jamk.fi

Mobile game prototyping

Programmer's point of view

Pasi Vilppola

Bachelor's thesis
April 2015

Degree Programme in Software Engineering
The School of Technology, Communication and Transport

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

Description

Author	Type of publication	Date
Vilppola, Pasi	Bachelor's thesis	22.04.2015
		Language of publication: English
	Number of pages 73	Permission for web publication: x

Title of publication
Mobile game prototyping
Programmer's point of view

Degree programme
Software Engineering

Tutors
Huotari, Jouni and Väänänen, Olli

Assigned by
Perttola, Jussi - Zaibatsu Interactive

Abstract

This task was assigned by Zaibatsu Interactive with the aim to create mobile game prototypes of an existing board game and to see how well the idea in the board game could be used in the mobile platforms and how could this basic idea be developed further. The project was carried out together with another student, Ville Hyytiäinen.

Unity is a game development engine used by more than four million registered developers around the world. It has five main views essential to the workflow: Project Browser, Hierarchy view, Game view, Scene view, and inspector. In addition to Unity the MonoDevelop tool is used for the scripting and it comes with Unity.

Prototyping is a way of testing ideas in practice and figuring out what works and does not work without the need to start building the product right away. Physical prototyping uses ordinary household objects such as paper and pen to test game ideas. Digital prototyping is a way of testing the ideas in their intended format. Small game mechanics should be tested with digital prototyping for example to find out how the controls feel.

Three game prototypes were created during the project. The first one was a straight up port of the original board game. The second was an evolvement from the original idea with new twists and the last one was a 3D game that had only the basic idea of the original board game.

The project was an immensely fun, interesting and at times challenging to be a part of. Unity proved to be an excellent tool especially for the beginners and that the same basic idea can be used to create very different games.

Keywords/tags
Game, Games, Prototype, Prototyping, Mobile, Programming

Miscellaneous

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

Kuvailulehti

Tekijä	Julkaisun laji	Päivämäärä
Vilppola, Pasi	Opinnäytetyö	22.04.2015
	Sivumäärä 73	Julkaisun kieli Englanti
		Verkkojulkaisulupa myönnetty: x

Työn nimi
Mobile game prototyping
Programmer's point of view

Koulutusohjelma
Ohjelmistotekniikka

Työn ohjaajat
Huotari, Jouni ja Väänänen, Olli

Toimeksiantaja
Perttola, Jussi - Zaibatsu Interactive

Tiivistelmä

Opinnäytetyössä oli tarkoituksena selvittää, kuinka tavallinen lautapeli-idea kääntyy mobiilialustoille ja mitä uutta tämä alusta voisi peli-idealle tarjota.

Unity on pelinkehitystyökalu, jota käyttää yli neljä miljoonaa rekisteröitynyttä käyttäjää maailmassa. Siinä on viisi perusosaa, jotka ovet välttämättömiä Unityllä työskentelyn kannalta: Projekti-selain, Hierarki-näkymä, Kohtaus-näkymä, Peli-näkymä, sekä Tarkkailija-näkymä. Unityn lisäksi MonoDevelop-työkalua käytetään koodin kirjoittamiseen. MonoDevelop tulee vakiona Unity asennuksen mukana. Tässä projektissa käytettiin Unity 4.6, sekä 5.0 versioita.

Prototyyppaus on tapa testata ideoita ennen niiden varsinaista toteuttamista pienin kustannuksin. Fyysisessä prototyyppauksessa käytetään mitä tahansa kädenulottuvilla olevia tarvikkeita, esimerkiksi tavallisia kodintarvikkeita, idean toimivuuden testaukseen. Digitaalinen prototyyppaus on idean kokeilua ja testausta alustalla, jolla tuote on tarkoitus kehittää, tässä tapauksessa Unityllä.

Projektin aikana luotiin kolme prototyyppi peliä. Ensimmäinen peli oli suora kopio alkuperäisestä lautapelistä, toinen oli peli, jossa ideaa vietiin pidemmälle ja siihen lisättiin uusia ideoita, ja kolmas oli 3D peli, jossa vain ihan perus idea pysyi samana alkuperäiseen nähden.

Projekti osoitti, että Unity on erittäin hyvä työkalu varsinkin vasta-alkavalle pelinkehittäjälle ja, että hyvin yksinkertaisesta perusideasta voidaan luoda hyvin erilaisia mobiilipelejä.

Avainsanat
Peli, Pelit, Prototyyppi, Mobiili, Ohjelmointi

Muut tiedot

Contents

Figures

Terminology

2D game

Two-dimensional game refers to a game type that restricts player movement to two dimensions such as moving to the left and right and jumping up and down in X and Y axis (2D (Concept) – Giant Bomb, N.d.).

3D game

Three-dimensional game refers to a game type that allows players to move in X, Y and Z axes in the game world such as car games where players drive forward, can turn left and right and jump up and down when driving over bumps in the game world (Slick, N.d.).

Android

Mobile operating system used by companies such as Samsung, Google, HTC on their phones and tablets (Android, the world's most popular mobile platform, N.d.).

Baldur's Gate

It is a fantasy role-playing game developed by BioWare and published by Interplay Entertainment in 1998 (Baldur's Gate –Wikipedia 2015).

C#

C# (pronounced C-Sharp) is an object-oriented programming language developed by Microsoft Corporation that can be used by programmers to develop computer software. It is based on C++ programming language and has similarities with Java programming language. (Rouse, 2007.)

Collider

Collider is a component in Unity that resamples the shape of a GameObject, is invisible and is used to detect physical collisions between GameObjects in the game world (Unity – Manual: Colliders, N.d.).

Deus Ex

It is a first person shooter, stealth and role-playing game developed by Ion Storm and published by Eidos Interactive in 2000 (Deus Ex – Wikipedia, 2015).

Diablo III

Diablo III is an action role-playing game developed and published by Blizzard Entertainment in 2012 (Diablo III –Wikipedia, 2015).

Dungeons and Dragons

“Dungeons and Dragons is a fantasy tabletop role-playing game designed by Gary Gygax and Dave Arneson and first published in 1974 by Tactical Studies Rules, Inc. The game has been published by Wizards of the Coast since 1997.” (Dungeons and Dragons – Wikipedia, 2015)

Dropbox

Dropbox is a service that can be used to easily share photos, videos and documents with others (Dropbox - About Dropbox, N.d.).

Epic Mickey

It is a platforming game designed by Warren Spector, developed by Junction Point Studios and published by Disney Interactive Studios Nintendo in 2010 (Epic Mickey – Wikipedia, 2015).

Flowdock

Flowdock is a tool that team members can use to communicate with each other. It is designed so that teamwork is a seamless and highly productive experience. (About Flowdock, N.d.)

FPS

First-person shooter or FPS for short is a term used for a type of action videogame genre that uses first-person perspective, in other words viewpoint of the protagonist to show a gun which player uses to shoot enemies in the game world (Janssen, N.d.).

GameObject

Every object in Unity is a GameObject. GameObjects are empty boxes that contain different kinds of components to create the character or a wall of the house in the game. (Unity – Manual: GameObjects N.d.)

Gaming platform

Platform refers to a hardware and software combination used to run other applications (Janssen, N.d.). Gaming platform is the hardware and software used to run games. An example of such a platform is PlayStation 4.

Git

Git is a version control system designed to handle everything from a very small to a very large software development projects (Git, N.d.). It has a directory that changes as new code and files are added or modified in an application development (Janssen. N.d.).

IDE

Integrated Development Environment or Interactive Development Environment is a program or a group of programs used by programmers to create computer software in software development (Janssen, N.d.).

IOS

Mobile operating system used by Apple in their devices such as iPhones, iPads and iPods (Apple - IOS8 - What is IOS? N.d.).

iPad

iPad is a tablet computer made by Apple (Apple – iPad, N.d.).

iPhone

iPhone is a smartphone made by Apple (Apple – iPhone, N.d.).

iPod

iPod is a music player made by Apple (Apple . iPod, N.d.).

Katamari Damacy

It is a third-person puzzle-action game developed and published by Namco in 2004 (Katamari Damacy – Wikipedia, 2014).

Pikachu

Pikachu is a fictional character from the Pokémon series owned by Nintendo Co., Ltd (Pikachu – Wikipedia, 2015.).

PlayStation 4

Video game system manufactured by Sony Corporation that customers can use to play games, watch movies and listen to music among other such things (PS4 – Features, Games and Videos, N.d.).

Prefab

Prefab is an asset type that stores GameObject object complete with components and properties. It can be used as a template for an object that is added in the scene later. (Unity – Manual: Prefabs N.d.)

Raycast

Raycast is like a laser beam that goes from one point to a specified direction in a game scene in Unity (Unity – Scripting API: Physics2D.Raycast, N.d.).

Rigidbody

A Rigidbody is the main component that allows physical behavior in GameObjects in Unity. Rigidbody makes the objects be effected by gravity and if any colliders are attached to the same GameObject Rigidbody handles the movement if collisions occur between objects. (Unity – Manual: Rigidbodies, N.d.)

Script

Scripts are like a set of instructions that the game engine can use to do different kinds of tasks. For example, a script might make an alligator rush out at a player or a light switch to turn on the lights. (Blackman 2014, 63)

Spore

Spore is a multi-genre fantasy god game designed by Will Wright, developed by Maxis and published by Electronic Arts in 2008 (Spore – Wikipedia, 2015).

Tag

Tag is a word that is linked to a GameObject which can be used to easily access that GameObject's properties. For example character in the game that the player of the game is controlling could have a tag named "Player". (Unity – Manual: Tags, N.d.)

Texture

"The visual and especially tactile quality of a surface" (Dictionary.com N.d.).

Trello

Trello is a tool that helps users to organize their projects into boards and cards. At a glance Trello can tell user what is worked on, by whom and in which state the process is currently on. (What is Trello? – Trello Help, 2015.)

Visual Studio Express

Visual Studio Express is an IDE, a free version of Visual Studio. It is considered to be a learning tool for example students. (Janssen, N.d.)

Xbox One

Video game system manufactured by Microsoft Corporation that customers can use to play games, watch movies and listen to music (Xbox One – Official Site, N.d.).

1 INTRODUCTION

1.1 Background to study

This task that was assigned to the writer of this thesis by Zaibatsu Interactive was to prototype a normal board game idea into a mobile platform using Unity game engine. The idea was to first test and see how well the basic game idea translated into a mobile platform. The next step was to start thinking what possibilities the mobile platforms offered, what changes could be made, to enhance the original idea and make it work better in that form. The task was to make several prototype games each playing differently but use the same basic concept that the original board game has.

The project was carried out by working together with Ville Hyytiäinen, where the work was divided into two sections: game design and game programming. Ville's responsibilities lay in the game design aspects of the game while the author of this thesis was tasked with programming the game. This division was made because this project was to resample a real world game development project.

The project started with the creation of the three working prototypes in collaboration with Ville Hyytiäinen. This was done by working together for four days a week, doing six-hour days for two and a half months. All the decisions concerning the project were made together and all the work was done in the same room. When the prototypes were finished at the end of March 2015 both members of the project started writing their own thesis. One afternoon was used to trade ideas on how to write the introduction to each thesis and each member of the project did rest of the writing individually. The thesis written by Ville Hyytiäinen will be submitted to Theseus by the end of May 2015 (Hyytiäinen 2015).

The project started out by testing the original game (which is described in more detail later) to find out what the basic gameplay elements where and what were the features that made the game fun in its original form.

1.2 Zaibatsu Interactive

Zaibatsu Interactive is a small indie game developer based in Finland. The company was founded in the spring of 2014 and the developers are working on their first game, Elder Goo (Zaibatsu Interactive – About Us.)

Zaibatsu Interactive was founded by a group of six people including CEO Jussi Perttola from whom this assignment originated. Their goal is to create games that are fun and engaging even after hours of gameplay. They also want to bring the essence of classic video games to mobile platforms (mt.).

Elder Goo is a puzzle-adventure game that can be played either by one player or with up to four friends. Players control four slimy characters through levels solving puzzles (Zaibatsu Interactive – Elder Goo. N.d.).

1.3 Motivation

The major motivation for me to start this project was that I have been a gamer all my life, I applied to JAMK University of Applied Sciences because I wanted to make videogames and I feel like programming is both challenging and rewarding; challenging because you never know at the beginning of a project, what problems you might face and you very much always face new ones. It is also rewarding because you can easily see at a glance what you have created.

It is also a huge motivation for me as a gamer to be a part of creating those worlds and experiences that I have personally enjoyed my whole life. I want to

be able to give that same feeling to other players with my own games someday.

The future of game development in Finland looks quite bright according to a study made by Tekes, which, of course, adds to the writer's personal motivation. As this study indicates, the Finnish game industry has grown from non-existent to the major content export industry in Finland. (Nordgren, 2014.)

2 GAME IDEA AND TECHNOLOGIES

2.1 Pikachu card game

The starting point for the prototyping process was the analogue Pikachu card game. This game was used as the basis of the first prototype on mobile devices. The game is a two- to four-player game. Figure 1 presents the Pikachu game that was used as the basis for this project.

Figure 1. The Pikachu game.

2.1.1 Contents of the game

This Pikachu card game contains a card deck of 27 playing cards, each card portraying a slightly different image of the Pikachu character and four sticks with suction cup at the end that represent the Pikachu's tail. The game also

contains a machine with 3 rotating plates and a button on top to rotate the plates. Figure 2 shows the contents of the Pikachu game.

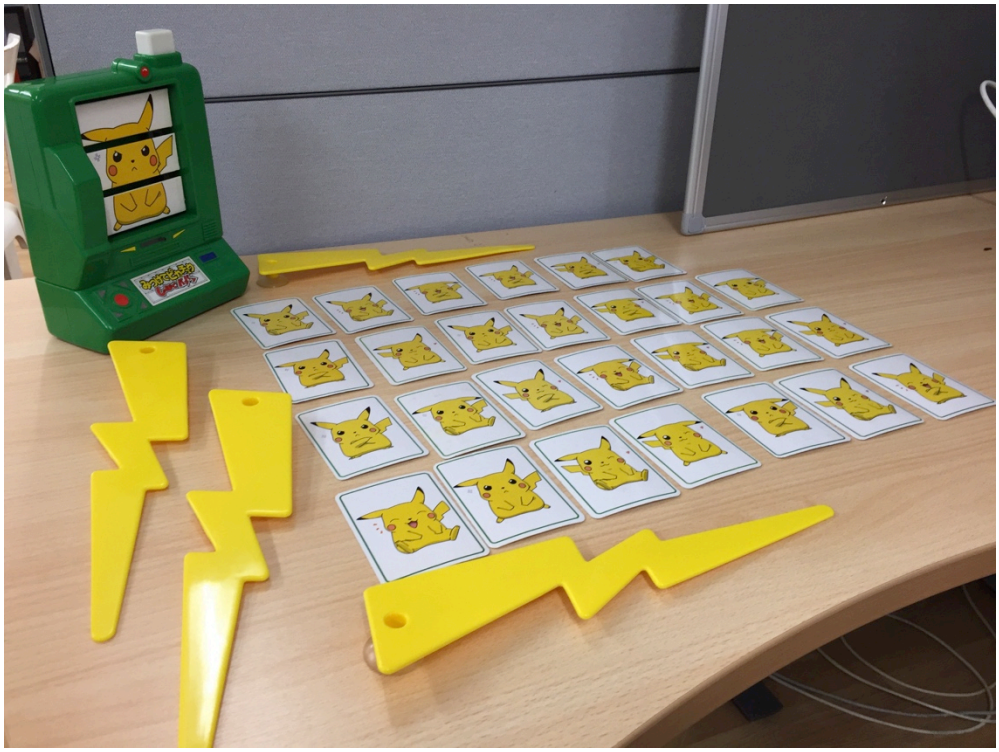

Figure 2. The contents of the Pikachu game.

2.1.2 Setup

The setup starts out with laying out the Pikachu cards face up on the table. Players can choose the layout themselves. Some might prefer layouts that have the cards setup neatly in even rows and columns and others might like to pile and mix them up.

After cards are laid out the way players want, each of the players is given one of the sticks with a suction cup at the end and the machine with the rotating plates is put on the table so that each player can see the generated picture as well as reach the button on top of the machine. After this the game is ready to start.

2.1.3 Gameplay

The game starts with players choosing the starting player who pushes the button on top of the Pikachu machine. This makes the machine start rotating the plates, each containing one partition of the picture to be generated. When the plates stop rotating the picture of the Pikachu character is formed and the players start looking for the matching picture among the cards laid on the table. Figure 3 presents the machine that generates the picture.

Figure 3. The picture generator.

The players now look for the right card and when a player finds the right card he or she then slaps the Pikachu tail stick on the table, hitting the card with the suction cup at the end of the tail hence picking up the card and moving it in front of him or her in the table. Figure 4 shows the Pikachu's tails. This card should still be in the reach of other players because when the game progresses other players can try to steal this card.

Figure 4. The Pikachu's tails.

Now the game moves on and another player pushes the button on top of the Pikachu machine and again the players try to find the card with the matching picture. Now this can of course be in the center of the table or in front of another player. This process is repeated until one of the players has five cards collected in front of them in which case the player with the five collected cards wins the game. Figure 5 shows three examples of Pikachu cards that players look for.

Figure 5. The examples of the Pikachu cards.

2.1.4 Principles used in prototypes

Thus the basic idea of the analogue Pikachu game is to look at the given example and find the matching picture faster than any other player. This was also the basic idea used in the prototypes created. Three full game prototypes were created and all of the versions created had the same basic principles in mind: an example to look for and objects to try and find, be it pictures, bugs or 3D objects in a game world. The goal was not to get hung up on the prototypes all being the same as the original board game but to try having a game that was completely different from the original game but still have the same basic idea.

2.2 Tools

Tools used in the project where mostly predefined by Zaibatsu Interactive. The tools were the ones that Zaibatsu Interactive staff use in their own projects;

therefore it was easier to stick to those existing tools than to look for others suited for the same project.

Development tools

As mentioned earlier, Unity game development engine was the main tool used to make the game prototypes. This is where the different scenes and GameObjects are created and managed.

The second most used tool was the MonoDevelop IDE which was used for scripting using C# programming language. This tool is tightly used with Unity editor and is the main tool for programmers working in Unity projects.

For version control used was Git, which is very fast and lightweight version control system.

Documentation and communication

For documentation, task management and communication Trello, Flowdock and Dropbox were used. Trello was used for assigning tasks, making bug reports and generally keeping track of which parts of the project each member was currently working on. It also worked as documentation on what had already been done. Flowdock was used for general communication between the team members and Dropbox handled small document exchanges between the team members.

3 UNITY

3.1 What is Unity?

Unity is a complete game development platform for creating complex 2D and 3D games. Game developers can use Unity free of charge to develop games from simplest 2D games to the most complex, state of the art 3D games available. (Unity – Game engine, tools and multiplatform, N.d.)

Unity is used by more than 4 million game developers (registered users at the beginning of 2015) around the world as can be seen in Figure 6 below. It has a very large and active community support that developers can use to get answers to problems they might face when building their own games. (Unity - Fast Facts. N.d.)

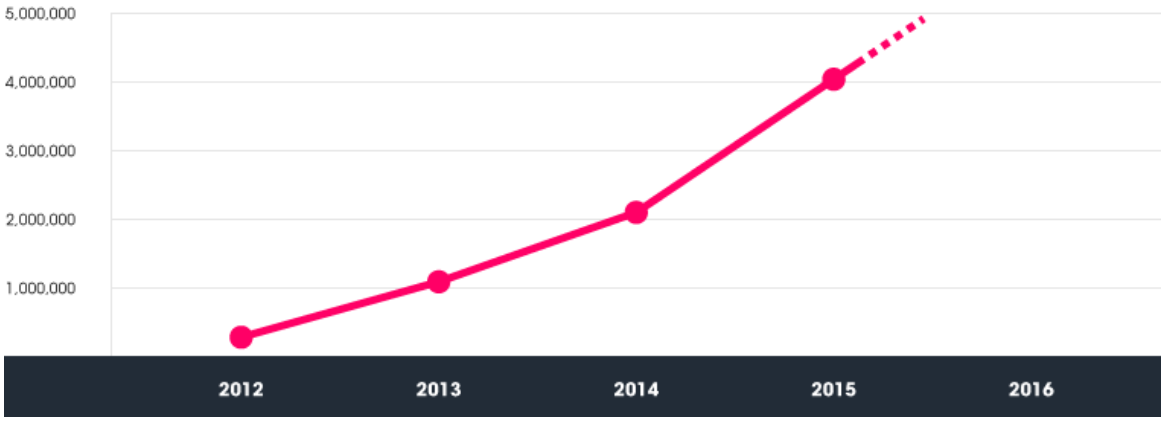

Figure 6. Registered Unity developers 2012-2015.

There are 21 different gaming platforms that developers can choose from when building their games in Unity most notable mobile platforms such as IOS and Android and console platforms like PlayStation 4 and Xbox One (Unity – Game engine, tools and multiplatform, N.d.).

3.2 Why Unity?

Unity Engine was used in this project because Zaibatsu Interactive uses it in their own game development projects. This makes it easier for them to use the finished prototypes in their possible future projects. Using Unity also made sense from the tutorial standpoint. It was easy to get assistance from the Zaibatsu Interactive staff if there ever was such a need since the programmers at the office had been using Unity for quite some time and because of that had a vast knowledge of the engine. Unity versions used in the project were Unity 4.6 for the first two prototypes and 5.0 for the final prototype.

Unity was also the most logical choice for this project because of its ease of use, vast amount of community support when searching for tips and answers to problems and the writers previous knowledge of the engine (Blackman 2014, xxxvi).

3.3 An Overview of the Unity Engine

This chapter discusses the Unity Editors main functionalities in some detail. Unity editor consists of five different parts or views that each have their own purpose in developing games with Unity. These views are: Scene, Game, Hierarchy, Project and Inspector. (Unity – Manual: Learning the Interface, N.d.) In addition to views, MonoDevelop is used closely with Unity editor for scripting purposes (Unity – Manual: MonoDevelop, N.d.). In Windows, Visual Studio Express can replace MonoDevelop as a scripting tool (Unity – Manual: Visual Studio C# Integration, N.d.). Figure 7 illustrates the default, empty, Unity project.

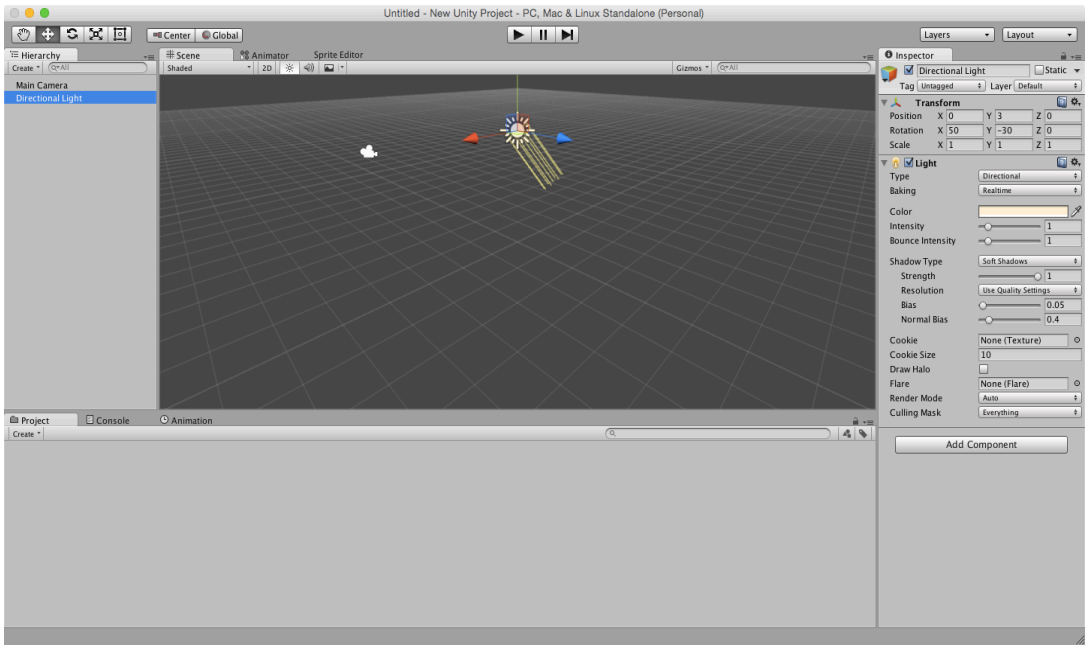

Figure 7. Default window of the Unity Editor.

3.3.1 Project Browser

Project Browser is where user can find all the assets associated with the current project. This can include among others scripts, textures, audio, prefabs and also the scenes or levels created and saved for this project (Menard 2012, 16).

Project Browser is a mirror of the assets folder in the directory where the current project resides. Assets can be searched using the search bar in the upper right corner of the Project Browser. Files can also be opened straight from Unity using the Project Browser just by double clicking the file to open and edit in the default application associated for that file. Scripts for example will be opened in MonoDevelop by default. After completing the necessary modifications and you saving the file, Unity will then reimport the file automatically so that the files are always up-to-date in the Project Browser. (ibid., 17.)

New assets can be created using the Create menu within the Project Browser. Simple clicking on Create, choosing what is wanted to create and Unity automatically creates that asset. All sorts of assets can be created from the Create menu like empty scripts, new folders or materials. (ibid., 17.) It is a good practice to create folders for different types of assets to keep the Assets folder and Project Browser neatly organized. Figure 8 below shows an empty view of the Project Browser.

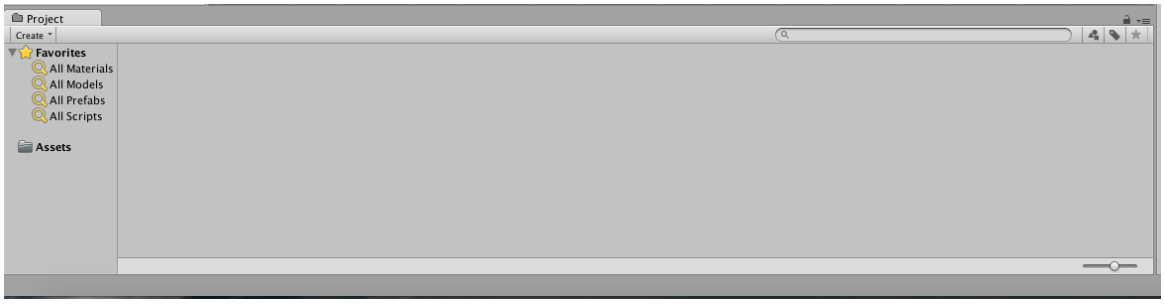

Figure 8. Project Browser in Unity.

The left side of the Project Browser shows two folders, Favorites and Assets. The assets are the Assets folder in the project directory as mentioned before and the Favorites are where user can add assets that are used mostly for quick access. (Unity – Manual: Project Browser N.d.)

Project Browser view can also be switched from 2-column view, which is the default view to one column view from the Window Options button in the upper right corner of the Project Browser. This essentially makes the Project Browser show only the hierarchy structure list without the favorite assets. (ibid.)

3.3.2 Hierarchy

Hierarchy view lists all the GameObjects used in the current scene. Naming the objects is quite important in the Hierarchy view since naming every single

cube object as a “Cube” makes it extremely difficult to find the right one later on. (Menard 2012, 19.)

Deleting objects in a Scene view will delete that object from the Hierarchy view but not, of course from the Project Browser. Also, deleting an object from the Hierarchy will delete the object from the Scene view. (ibid. 19.) Figure 9 presents the Hierarchy in Unity.

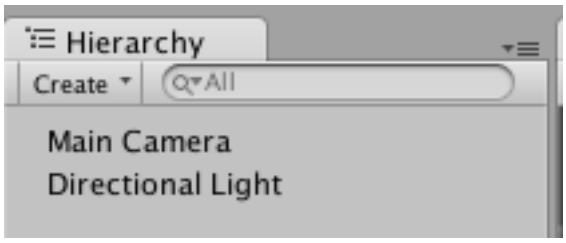

Figure 9. Hierachy in Unity.

GameObjects can be organized by parenting some GameObjects with others. This is implemented similar to folders in a directory. Choose a folder (parent GameObject) and drag a file (child GameObject) inside that folder. Parenting is quite useful for moving a large amount of objects around in the Scene view. (ibid., 20.)

3.3.3 Toolbar

The toolbar consists of five basic control groups for the game. These groups are: (Menard 2012, 23.) Figure 10 illustrates a Unity toolbar.

Transform tools

Transform Gizmo toggles

Play controls

Layer drop-down

Layout drop-down

Figure 10. Toolbar in Unity.

Transform tools are used to manipulate the GameObjects in the Scene view. The first one from the left is a hand tool for panning the scene around. It is used for getting a better view of a point of the scene you want to work on. The second tool from the left is the translate tool used to move GameObjects around in a Scene view. Middle tool in the Transform tools is Rotate tool used to rotating GameObjects in a Scene view. The second tool from the right is the Scale tool and it is used for scaling of the objects. (ibid., 23.) The right most tool is the Rect tool, which is used to view and edit Rect handles on a 2D and 3D GameObjects (Unity – Manual: Toolbar N.d.).

Next to Transform tools are the Transform gizmo toggles. These toggles are used to transform objects from different coordinate systems and pivot points. (Blackman 2013, 32.)

Center controls in the toolbar are the Play controls. These are used to run the current scene in real time to see how everything works as if the developer was the player. These controls contain Play, Pause, and Skip buttons. (ibid., 32)

Last two tools in the toolbar are the Layer and Layout drop-down tools. The layer tool lets the developer to choose which layers in the Scene view are currently visible and the Layout drop-down is used for choosing between different layout options for how the views are shown in Unity. (Menard 2012, 23.)

3.3.4 Scene View

Scene view is one of the most important views in Unity. It shows every GameObject in the game and it is used to layout the levels the way they are designed. Every object from the hierarchy view is shown here and they can be

moved, rotated scaled etc. right from the Scene view. (Menard 2012, 23.) Empty Scene view can be seen in Figure 11 below.

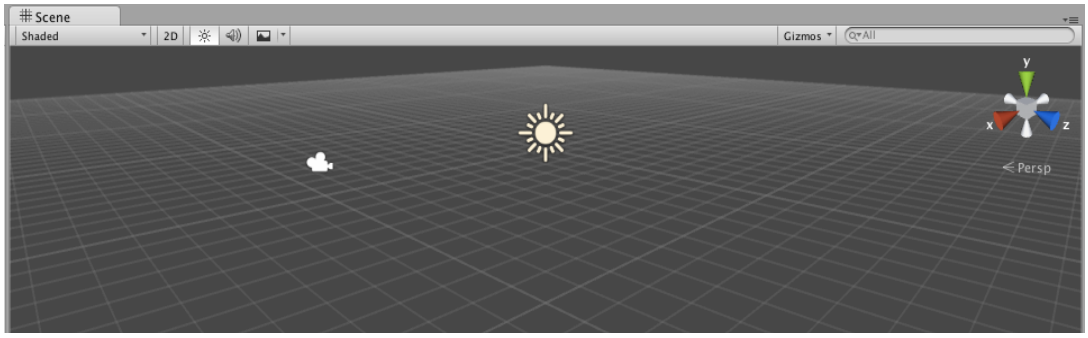

Figure 11. Scene view in Unity.

The objects in the Scene view can be manipulated using the Transform tools mentioned in the Toolbar section. Simply clicking a GameObject makes it editable. After this the object can be rotated, moved and scaled using the Transform tool and the gizmos that show up in the Scene view as seen in Figure 12.

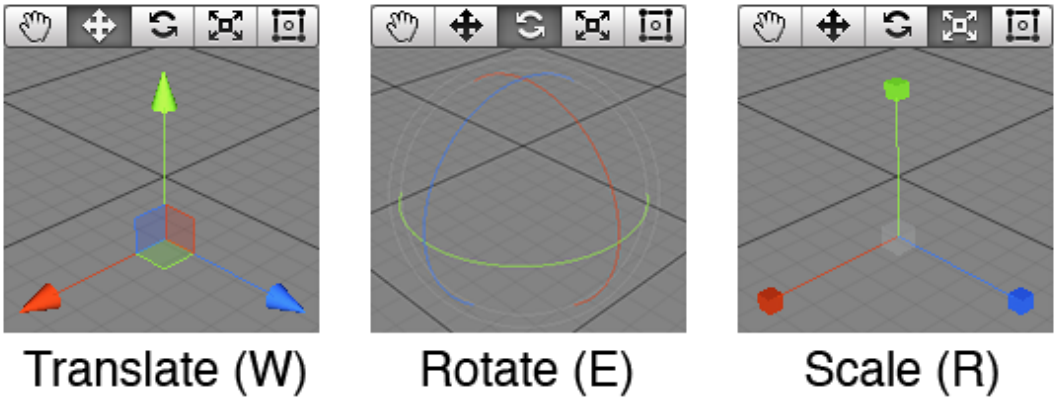

Figure 12. Move, Rotate and Scale gizmos.

The scene view also contains a tool called Scene gizmo shown in Figure 13. This gizmo can be used to change the point of view in the Scene view. By clicking on the axis cones in the gizmo the view turns to show the scene from that particular direction.

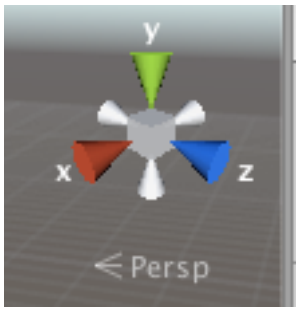

Figure 13. Scene gizmo.

The viewing perspective can be changed between the perspective view and isometric view by clicking the text under the gizmo. This can be extremely important when fine-tuning small aspects in GameObjects for example.

3.3.5 Game View

Game view is the view used for previewing the game. In this view the game is shown exactly as the player would see it and the game can be tested to see how it plays. It can be started straight from the editor using the play button. (Menard 2012, 36.) Empty Game view window can be seen below in Figure 14.

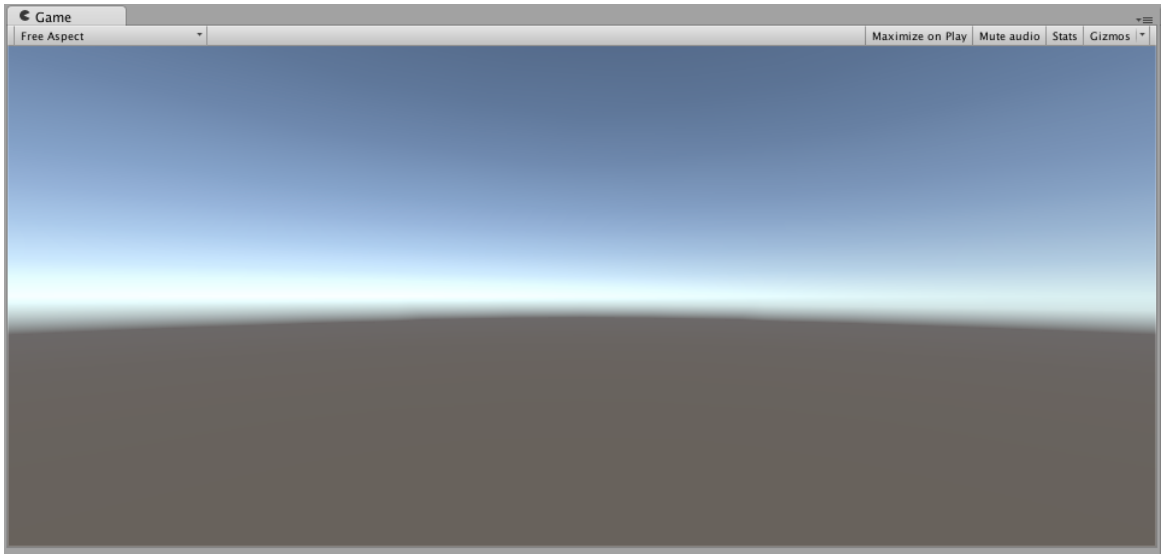

Figure 14. Game view in Unity.

When running the game in Unity, it is good to remember that any changes made to GameObjects during this time will not be saved. Changing the values of a GameObject when the game is running, however, can be useful when testing certain parts of the game. For example it could be tested how high the character jumps by changing the jumping force while running the game. (Menard 2012, 36-37.)

Game view has its own Control bar, which can be used to adjust various options. First in the Control bar there is the Aspect selector. With this the aspect ratio used to render the game in Game view can be adjusted. Options that can be selected from this menu depend on the platform chosen to build the game on, which in turn can be adjusted from the Build Settings in Unity. Second in the Control bar is the Maximize on Play button used to maximize the Game view window when the Play button is pressed. Last two controls are the Gizmos selector and the stats button. Gizmos selector lets users choose which gizmos from the Scene view are shown in the Game view. Stats button shows different kind of statistics of the game that's running and can become useful when optimizing the game. (ibid., 38.)

3.3.6 Inspector

Inspector view is where all the Components attached to the selected GameObject such as scripts, colliders and animators can be seen. Alongside the Scene view, this is one of the most used views in Unity. This is where the character is made to run or enemies are made to shoot at a player. Inspector view can be seen in Figure 15 below. In this Figure the Directional light object is selected and components attached to it are revealed in the inspector window. (Unity – Manual: Inspector N.d.)

Figure 15. Inspector in Unity.

The Inspector shows detailed information on the GameObject that is currently selected. This includes all the attached components and properties of that GameObject. All of these can be used to modify the GameObject's functionality. (ibid.)

In the Inspector underneath the GameObject name are drop-down menus for Tag and Layer. The tag is used to give GameObject a tag to easily find and

reference the GameObject in scripts later on. The layer can be set to the GameObject to for example make the camera not render certain layers in certain situations or to use in raycasting to allow the raycast to hit only certain layers. (ibid.)

The GameObject can also be deactivated from the Inspector by removing the tap from the left side of the GameObject name or the GameObject can be set static by adding the tap to the right side of the GameObjects name. (ibid.)

The details of the component attached to the GameObject can be hidden or shown using the arrows in the left side of the component name. On the right side of the component name is a question mark that can be used to go straight to the Unity reference page of that particular component. (ibid.)

There is also a small gear next to the question mark mentioned above. This can be used to reset the component values, copy the values to use with other component, remove the component from the GameObject or to move the component up or down in the Inspector window. (ibid.)

3.4 MonoDevelop

MonoDevelop is the default scripting IDE that comes with every download of the Unity engine. Scripts that are created in Unity and opened are opened in the MonoDevelop IDE. (Unity – Manual: MonoDevelop.) The default window of the MonoDevelop can be seen in Figure 16 below.

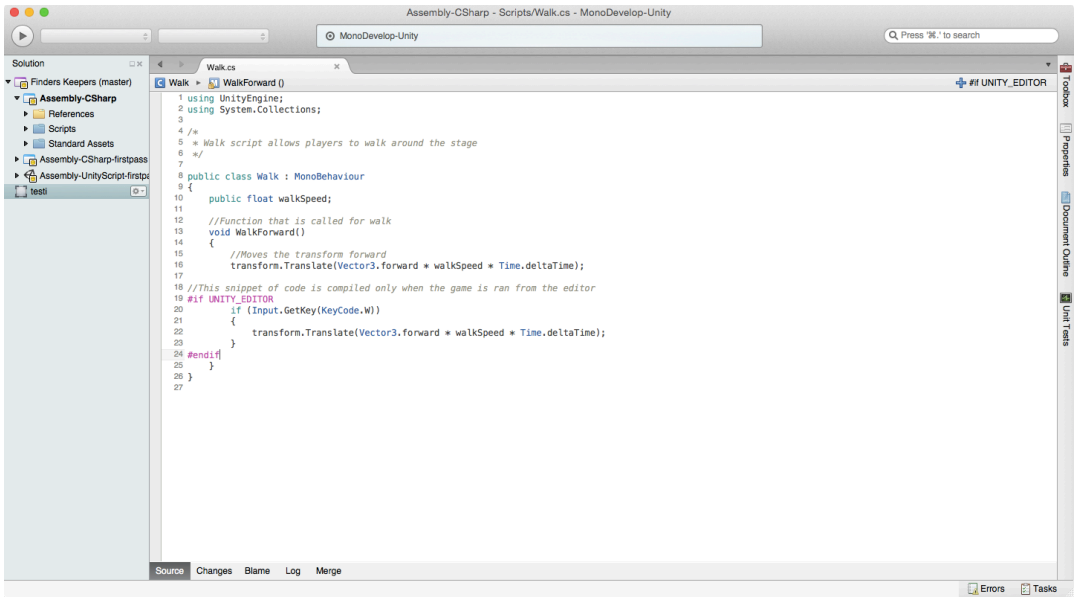

Figure 16. MonoDevelop IDE.

The left side of the MonoDevelop window shows the same hierarchy that is used in the Unity Assets folder. Double clicking a script opens a new tab in the MonoDevelop for that script.

Once the developer has made all the necessary changes and additions to the script, the script is then saved by using the file menu or by using the hotkeys associated for saving (cmd + s in Mac, ctrl + s in Windows) and the developer returns back to the Unity to test that the script is working correctly by running the game or to continue working on the game.

4 PROTOTYPING IN VIDEO GAMES

4.1 Prototyping in general

The prototyping is an extremely important part of the video game development. It is a process where the developers create sort of a rough sketch of the game or a part of the game. The prototyping can be used to test out a new game mechanic or an idea that the developer thinks might be fun for the gamers. This allows the developer to test if an idea is as good in practice as it is as an idea on paper without spending too much time on creating artwork, audio and more. Prototypes are still playable of course. (Fullerton 2014, 197.)

A lot of times the developer might rush in to the making of the game without good prototyping of the idea first. This can prove to be a huge waste of time if the idea behind the game does not work in practice as it did as an idea because the developer has spent a lot of time making the game look good and sound right. In prototyping the developer does not need to be concerned about perfecting the look or about optimizing the technology but instead focus on that one aspect to see if it works. (ibid., 197)

There are a lot of different types of prototyping that can be used by the developers to create the prototypes for example, physical, visual, video and software prototypes. Those different types of prototyping could all be used by the developer in a single project if need be. The important thing to remember is that the developer should not try to prototype the whole design but only the aspects that they see needs prototyping. (ibid., 197) The next chapter discusses more about the physical prototyping.

4.2 Physical Prototyping

The physical prototyping is a form of prototyping where the developer uses all sorts of physical objects to create a rough design of the mechanics they want to prototype. These objects can be cards, cardboards, toy figures, pencils and paper to anything that the maker of the prototype might find useful in illustrating the mechanics prototyped. The physical prototypes are the easiest to make because the maker can use such ordinary things to create those prototypes. (Fullerton 2014, 197.)

There are many upsides to the physical prototyping. The first is the cost of the prototyping. It is very cheap to test out new ideas and mechanics when only things that the developers need for example, are everyday household objects. Another big advantage in physical prototyping is that the programmers might get attached to their code very easily and changing something later on can be a challenge where in the physical prototyping it is very easy to change how something operates just by rearranging the pieces. This makes trying different things and mechanics very fast. It is also easy to involve a nontechnical person from the team to prototype the game because that person will not need any programming experience to participate in the design process. (ibid., 198.)

Many of the most popular video games have also started out as board games. For example, popular games like Diablo III and Baldur's Gate are both derived from the board game version of the Dungeons and Dragons. As a matter of fact some of the game designers today have started their careers by making board games, one of these individuals being Warren Spector. (ibid., 200.) Warren Spector is the game designer on games such as Deus Ex and Epic Mickey (Warren Spector – Wikipedia 2015).

The physical prototype is not supposed to replace the digital prototyping says Fullerton (2014, 209), but to support it. The things that require real time testing

could still be done digitally for example, to see how a character moves through the game world. It is easy to use the physical prototyping to present the designers idea to a room full of programmers rather than just by trying to explain it. In the next chapter the author of this thesis will discuss the digital prototyping of video games.

4.3 Digital Prototyping

At some point in the making of the physical prototype it becomes clear that not all aspects of the game can be prototyped physically, especially if the game that is being developed is a video or a mobile game. This is where the digital prototype comes into play. With the digital prototype the developer can test aspects of the game such as how the controls feel or how a certain environment would act in a certain situation that could not be possible to get a feel of in the physical prototype. But making a digital prototype does not mean that the designing of the game has to be started from scratch. The physical prototype has already laid the foundation for the digital prototype so that the developer can extend from that. The digital prototype in other words lets the developer to test the ideas in the intended format. (Fullerton 2014, 235.)

According to Fullerton (2014, 235) the director for Spore, Eric Todd has divided the digital prototyping into four distinct categories: game mechanics, aesthetics, kinesthetic and technology. The digital prototyping should, like the physical prototyping, be done using minimal resources in the artwork, sound and such because the purpose is to test small portions of the game (ibid., 235.)

4.3.1 Game Mechanics

The game mechanic prototyping is something that if the physical prototype is done well the developer can be half way done with the digital prototype of a

game mechanic. But sometimes a digital prototype of a game mechanic is needed to fully get the feel of that particular mechanic. One of the important things to remember when making the game mechanic prototype is to focus on a simple question the developer has about the game mechanics. Later on the developer can test out the integration of the smaller digital prototypes to see how the prototypes work together but first it is important to focus on a single prototype and the game mechanic. (Fullerton 2014, 236.)

The example of the game mechanic prototyping in the digital prototyping comes from Jonathan Blow the independent game developer of the game Braid. Jonathan Blow used the digital prototyping to test out the time mechanics in Braid that allow players of the game to go back in time in an all-new way. Blow started the process by prototyping the idea of a billiard game where players could see the results of their shot before the shot was made. This did not turn out to be a very fun game but was very informative and eventually led to the time shift design of Braid. (ibid., 236.)

4.3.2 Aesthetics

The aesthetics are the visual elements of the game. The developer should not go in to a lot of detail in the aesthetics of the game in the digital prototyping phase but it is sometimes necessary to test out some aspects of the aesthetics too before starting to make the final look and feel of the game. These aspects can be things like how the combat works with the character animation or how the user interface fits the environment. The artist should still always keep in mind when the aesthetics are at the prototyping level and when too much time is spend in fine-tuning the look and feel of the game essentially missing the point of prototyping. (Fullerton 2014, 238.)

4.3.3 Kinesthetics

The kinesthetics is the feel of the game. It is how the game feels when the player starts playing it. It is how the controls feel while playing the game or how responsive the user interface (referred to as UI from now on in this thesis) is. The kinesthetics are practically impossible to prototype physically because the intended format is essentially the one that defines the controls of the game and how it feels to play the game in that particular platform. For example, it is completely different experience to play games on mobile platforms using the touch controls then it is on the PC using the mouse and keyboard controls.

The kinesthetic prototype can greatly help the developer to sell the game idea forward. Great example of this is the development of the game Katamari Damacy. The idea of the game is to roll a sticky ball around the game world and collect objects with it. This game idea does not sound all that fun in paper as it might be in reality so the creator of the game Keita Takahashi made a simple digital prototype of that basic game idea and convinced Namco to publish it for the PS2. The game used simple twin-stick controls that were fun and intuitive. (Fullerton 2014, 239-240.)

4.3.4 Technology

The technology is prototyped to find out what are the best tools to work with in a particular project. This is the way to ensure that the workflow in the project is as fluid and as efficient as it can be for the team. The technology prototyping can be done for example, on AI systems used, the physics engines or any number of different aspects of the game that require such tools. (Fullerton 2014, 241.)

Idea in the technology prototyping is to test out the tools with code and objects that the team is using in the game to see if the technology works for the

project. However it is not intended for example, to use the code created for the prototype in the final product as it is. This can be avoided for example, by using a different programming language in prototypes then what is used in the making of the final product. That way all the code must be written from the ground in in the actual product. (ibid., 241)

5 CREATED PROTOTYPES

5.1 Assignment

The work started on this thesis in December of 2014 with pitching of ideas at the Zaibatsu Interactive offices in Jyväskylä. The Zaibatsu Interactive CEO Jussi Perttola had a few ideas he suggested to Ville Hyytiäinen and the writer of this thesis about what the work could involve and the prototyping of an existing board game idea in to a mobile game was the immediate favorite for the whole group. This led to the decision that the prototyping of the game was going to be the subject for this assignment.

Next up was the play testing of the original board game to learn how it was played. A few hours were used to play the game while taking notes on things that made the game great in this platform and the aspects that should definitely be included in the mobile prototypes. These aspects were the example image that the board game generated randomly every round, the searching aspect where players looked for the card among many very similar cards and the stealing aspect that allowed the players to steal cards from other players. While play testing of the original game it was decided fairly quickly that the first prototype was going to be a straightforward port of the original game.

5.2 Card game

5.2.1 Physical prototyping

From the testing of the original game, the work moved on to light physical prototyping with pen and paper designing the first prototype. Paper was used

to replicate the screen of an iPad and small pieces of paper were cut in the shapes of small cards to represent the playing cards on the small screen. With this physical prototype it was immediately noticeable that the amount of cards that players could collect would have to be reduced but it was also noted that the accurate number of the cards that would fit the screen of a tablet computer would have to be tested digitally. This was so that it could be verified how clearly each cards picture could be recognized when the image of them got smaller to fit the tablet computers screen.

Various different layouts for the cards on screen were tested using physical prototyping. These layouts were hand-drawn and later on made better using the program Paint by Ville Hyytiäinen. A problem that arouse with the layouts of the cards were that how could the cards be laid out so that all of the players sitting around the tablet would see the cards the same way so it would be fair to each one. The layout was chosen to be the one that had the cards be sideways for everyone and the center of the screen to divide the other half of the cards to face right and other half of the cards to face left. This way all the cards were show to all the players in the same way. The final layout of the cards was chosen to be the one in Figure 17.

Figure 17. The final layout of the cards.

In addition to the layout of the cards, quick and simple card pictures created by Ville Hyytiäinen were used throughout the card game prototyping process. The example of the cards used can be seen in Figure 18.

Figure 18. The example of the cards used.

5.2.2 Starting the card game

Now it was time to fire up the Unity Engine and start creating the first prototype. The work with Unity started with the mechanics of dragging the cards on screen. This was first done using Unity's build in mouse events. The idea of the card dragging was that when player touches the screen a Raycast is generated from the point of the touch to the game world. If Raycast is hitting any of the cards that were movable the card is marked as picked up and in every update call of the game the position of that card is changed to the position of the touch that hit the card. Update is a MonoDevelop function that is called every time the screen is updated. This dragging of the cards using mouse events was a fairly simple process that was easy to implement in the prototype but it was not the method that could be used for long because it only worked for a single touch happening in the screen at a time.

5.2.3 Adding player slots

When the dragging of the cards was working the next step in the process was to get the cards to stay in the player card slots and to fall back to places they were picked up if they were dropped in wrong places. Of course the card picked up should only stay in players card slots if the card was the same as the example card shown to all the players. Otherwise the card should return to the place it was picked up from. The card slots referred here are the empty slots for the cards that each player had three of. This represented the aspect from the original game where each player kept the cards in the table in front of them when they had found the card the quickest.

There were a lot of difficult tasks involving adding the cards to player slots. It had to be checked which player's card slots were the dragged card touching and if the card was the right one and if the player dragging the card had let go of the card. And of course one of the biggest problems was that if the player already had a card that the new card added to player's card slots would not go over the previous card.

First, four GameObjects were created each representing one of the players card areas. Then collider and rigidbody components were added to each of these GameObjects. This way it could be determined that the card was actually on top of the player's area and should be added to the players card collection if it was dropped. Next, each of those colliders was given tags according to players using colors as indicators for each player. Finally a script was added to these GameObjects to keep track of the individual slots if they contained cards or not. This script had a Boolean value that indicated if a certain slot from a player had a card or if it was empty. This script was also used to save the card object in a list so it could be used later if needed.

5.2.4 Randomizing the game

Now that the players could drag cards to their card slots the example card randomizing was implemented. The basic view of the game had two images of the example card, one on each side of the screen facing different directions. These example cards can also be seen in Figure 17. These card images were randomized in the beginning of the game and each time one of the players in the game dragged a right card in to their card slot area. Randomizing was done in a way that a card index number (referred to as card id for the rest of the thesis) was selected randomly from between zero and a number of cards available in the game. This card id was then used to select the image sprite for the example card from the sprite sheet of card images. This card id was also used to check if the card that player dragged to their card slot area was the same one that the example card showed. In this stage the card id variable was also added to all of the cards in the table and each card was given a unique id so they could be recognized.

Now the card game prototype was working in it's basic form like it was intended excluding the fact that the table cards were in the same positions in each play through of the game. This was the time when the controls were changed from mouse event based controls to touch controls that allowed for four simultaneous touches on the screen. To implement this Unity's own Touch class were used. This system worked similarly to mouse event-based system used earlier but had the advantage of more then one simultaneous touch on screen. At first the touch-based system was used in multiple scripts at once and it turned out to be hard to keep track of from where any touch event was triggered at any given time. Later on in the project the use of the touch controls became clearer and a lot of refinements were made to its implementation.

At this point the table card randomizing was also implemented so that every time the game was restarted the table cards were in different order. This was done using the amount of card sprites in the card sprite sheet. There were 18 cards so the numbers from 1 to 18 were put into a list in a random order and then that list was used to add each of the table card GameObjects a sprite image from the card sprite sheet using the numbers in the list. In this way in each play through the cards were in random places in the table.

5.2.5 Finalizing the basic card game

Now it was time to wrap up the first prototype of the game by making menus to the game and going over the final bug fixes. Because the idea for this project was to make prototypes from the same idea, very little time was used on creating the menus. Just a basic main menu was created that had a few buttons for launching the game and a later on for launching the options menu. Unity's new UI tools were used in creating of the menu elements and each of the different menus had their own separate scene where they were build in. Figure 19 shows the first main menu iteration.

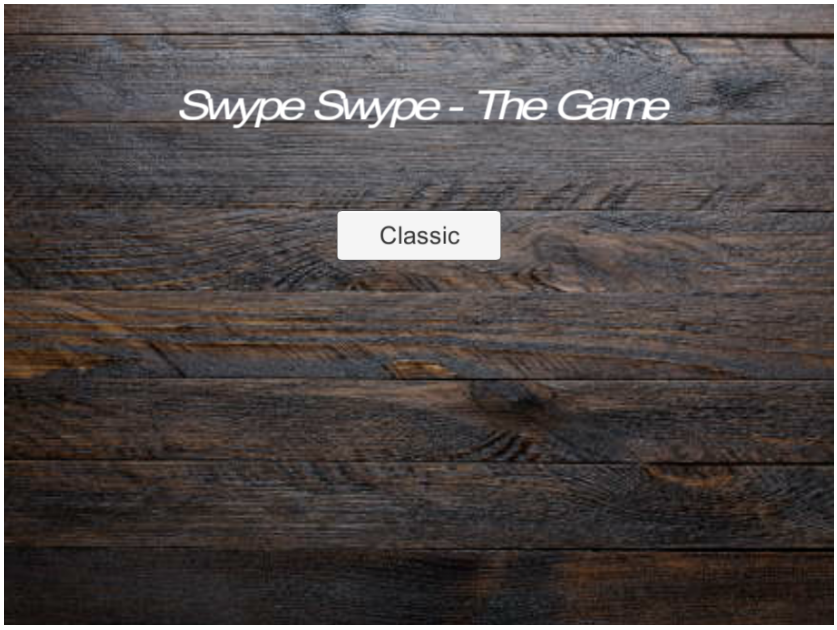

Figure 19. The first version of the main menu.

After the main menu a player select screen was build. This screen was used to let the players of the game to choose how many players would be playing. It was created so that players tapped the area on their side of the screen to confirm they wanted to play and if two or more players were selected the game could be started by tapping the play button in the middle of the screen. This selection would then determine the layout of the game screen and the player colors that would be in the game. In the first version of the player select screen a timer was used to start the game. The timer was set to ten seconds and if more then one player had been selected the game started automatically otherwise the timer restarted. After some time play testing the game it was decided to add the play button used to start the game manually and the timer was removed. This was found to be more intuitive and reduced the waiting time for the players to get in to the game. The first version of the player select screen can be seen in Figure 20 and the final version in Figure 21.

Figure 20. The first version of the player select screen.

Figure 21. The final version of the player select screen.

Next addition to the basic card game was the wait timer for the example cards. This was done to give players time to orientate to finding the next example card. What this timer did in practice was that before showing the new example card in the beginning of the game or after the last card was found the game would wait a few seconds and then reveal the new example card. An animation was used to flip the card over. In addition to this turn timer new layouts for the player areas were created to two and three player games. An example of the two player game layout is illustrated in Figure 22.

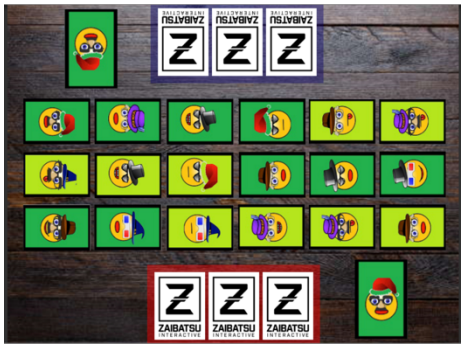

Figure 22. The two player layout.

Finally a game over and a pause screens were created to allow the players to start a new game or to return to the main menu. The pause and the game over screens were identical except for that the titles and the information texts were unique to each one. Figure 23 presents a pause screen on top of the basic card game.

Figure 23. The pause screen.

5.2.6 Adding bombs to the basic card game

When the basic card game was finished and the majority of the bugs were fixed it was time to add some flavor to the basic game. It was decided to add a new game mode that evolved the basic card game idea with bomb cards. A picture of the bomb card is shown in Figure 24.

Figure 24. The bomb card.

The idea for the bomb card was that it had a timer counting down from five and would explode when the timer runs out. Before this happens however the players are supposed to find the card from the table that looks like the bomb card and drag it to some other player. This in turn would make the player that received the bomb card to lose one of the cards. Then again if none of the players was able to find the card in time the bomb card would explode in the table and all of the players would lose a card they had previously collected.

When this bomb card mode was implemented in its basic form some options were added to vary the functionality of the bomb card. One of these options was to make the bomb card invisible at first and just when a player touches the right card it would be revealed to be the bomb card for the players. This added an extra layer of fun to the game because players might accidentally drag the bomb card to their own player area. Additionally an option was added to change the behavior of the bomb card exploding on the table. If none of the players could find the bomb card in time the bomb would explode and damage all players but this new option changed it so that the bomb exploding on the table would damage none of the players.

After this some new buttons were added to the main menu to allow players to choose which game mode to start and also an option menu to change various settings such as the behavior of the bomb card and how often the bomb card would appear in the game. Figure 25 shows the final main menu screen and Figure 26 presents an image of the options menu.

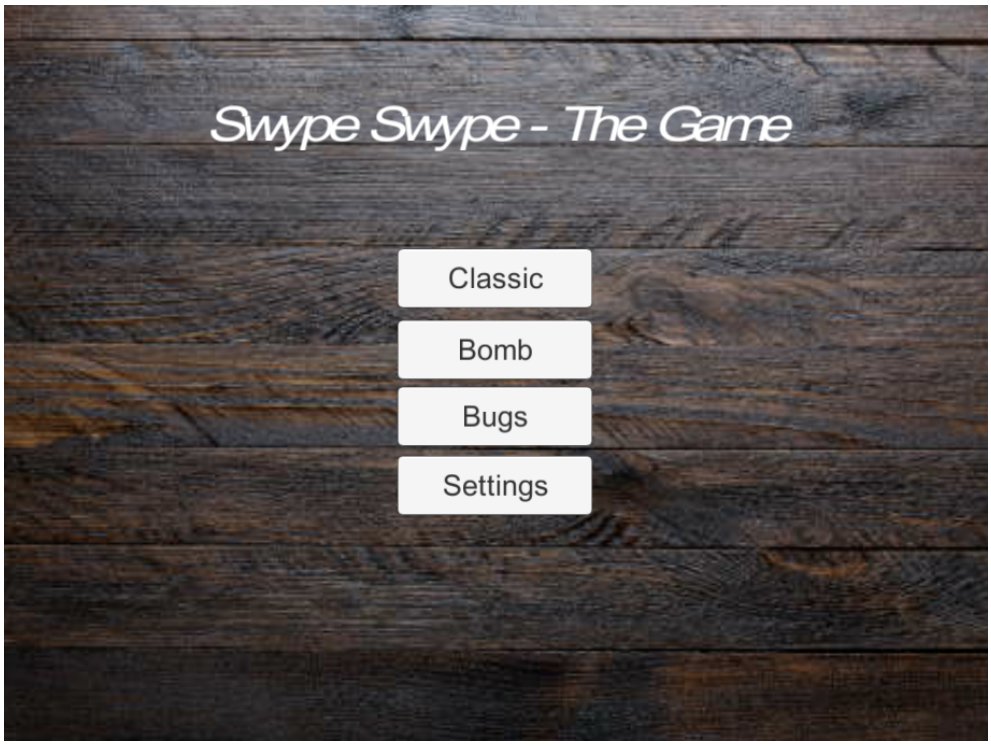

Figure 25. The final main menu screen for the card and bug games.

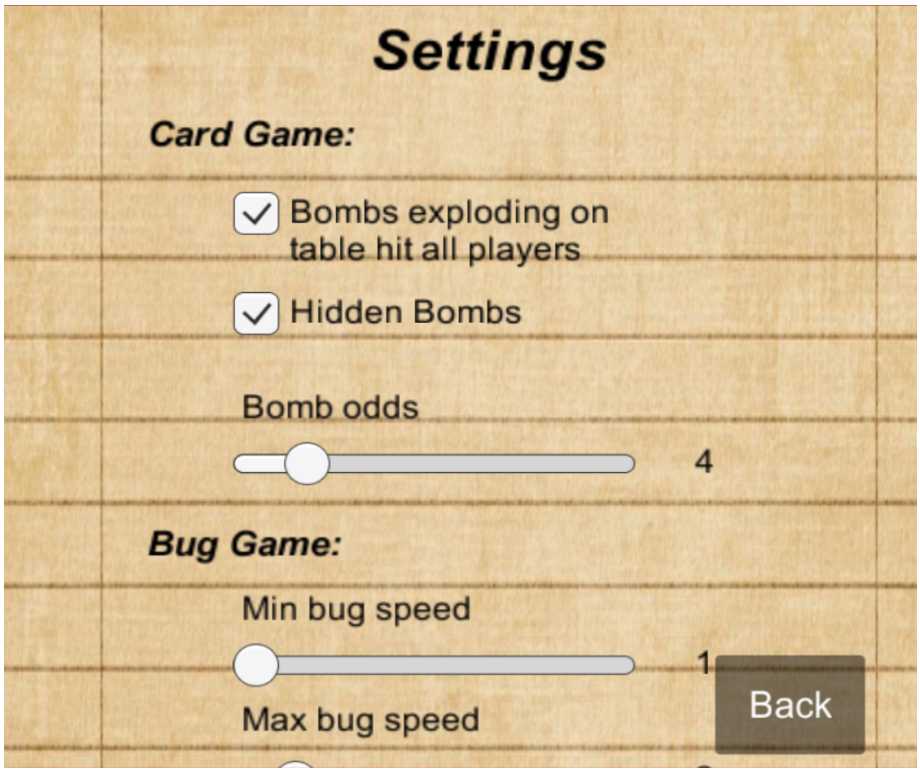

Figure 26. The options menu.

5.3 Bug squash

5.3.1 Moving on to the next iteration

Now that the card game prototype was finished it was time to move on to a next iteration of the same basic idea. There was a lot of ideas for the next prototype from additional modes to the card game to 3D Lego building games but eventually Ville Hyytiäinen and the writer of the thesis settled on the idea of a bug collecting game. It was chosen because it felt like the evolution of the card game and because of the mutual interest towards the idea. All of these ideas came from the minds of Ville Hyytiäinen, the writer of this thesis and the team at Zaibatsu Interactive.

The basic idea for the bug game was very close to the card game idea where players had to find the right bug among a lot of other bugs and then drag that bug to their own table of bugs. When one of the players collected five bugs they won the game. The stealing aspect of the original board game was removed from this iteration of the game to simplify the idea a little. Although the stealing aspect was removed the bugs were made to move around in the game area. This made the finding of the bugs a little trickier because they were constantly changing positions. Later on in the development of the bug game prototype some events were added to the game to add variety. These events are explained in more detail a little later.

5.3.2 Starting the bug game

The making of the bug game began with the movement of the bugs. The bug movement had to be random so that bugs would all walk in different direction on screen at different times. Also all the bugs had to have a little difference in their walking speeds. Also bugs should not go over the boarders of the game

screen or walk under any of the player table objects or the example bug space.

First adding a Rigidbody, a Collider, a Sprite renderer and a script component to the GameObject, created the bug GameObject. The sprite sheet of the original bugs created by Ville Hyytiäinen is illustrated in Figure 27. Next the bugs were made to walk in any direction using the Vector3.MoveTowards – function in Unity. What this function does is it takes the GameObjects current position and a target position specified by the developer and moves the object towards that target position with the speed that is defined by the developer. Then when the bug reached the target position a new target would be given to the bug and it would rotate and start walking towards that point in screen. Now that the basic movement was created for the bugs it was time to add other pieces to the game scene.

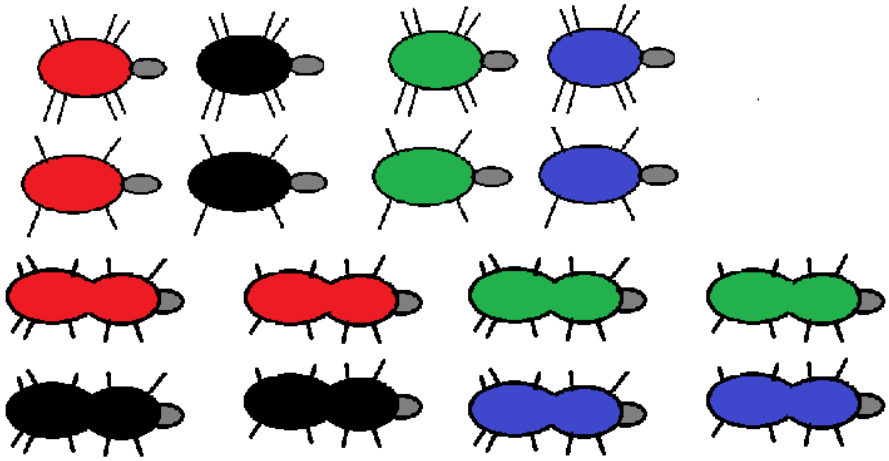

Figure 27. The original bug Sprite sheet.

5.3.3 Adding all the pieces to the bug game

For the example bug a see through dome was created and the bug was added inside the dome and made to rotate to give an impression of the bug floating in a water bowl. Then four player tables were created for the players to drag the bugs in. These tables were made to scale up in size when a bug was dragged

on top of it. This scaling was done using the Unity's Animator component. More bug GameObjects where also added to the game to see how all the GameObjects performed in the scene. It was soon discovered that collision between the bugs to each other had to be disabled because there was a lot of bugs in the scene at a time and using collisions the bugs got stuck in groups and would not walk around as intended. It was also noted at this time that the bugs should get a new target to walk towards when they collided to the dome containing the example bug. Otherwise the bugs would get stuck to the dome and again not function as was originally intended. Random targets that were used for the bugs to walk towards were also created from the area on screen so that they could not be under the player tables. In Figure 28 the first layout of the bug game can be seen.

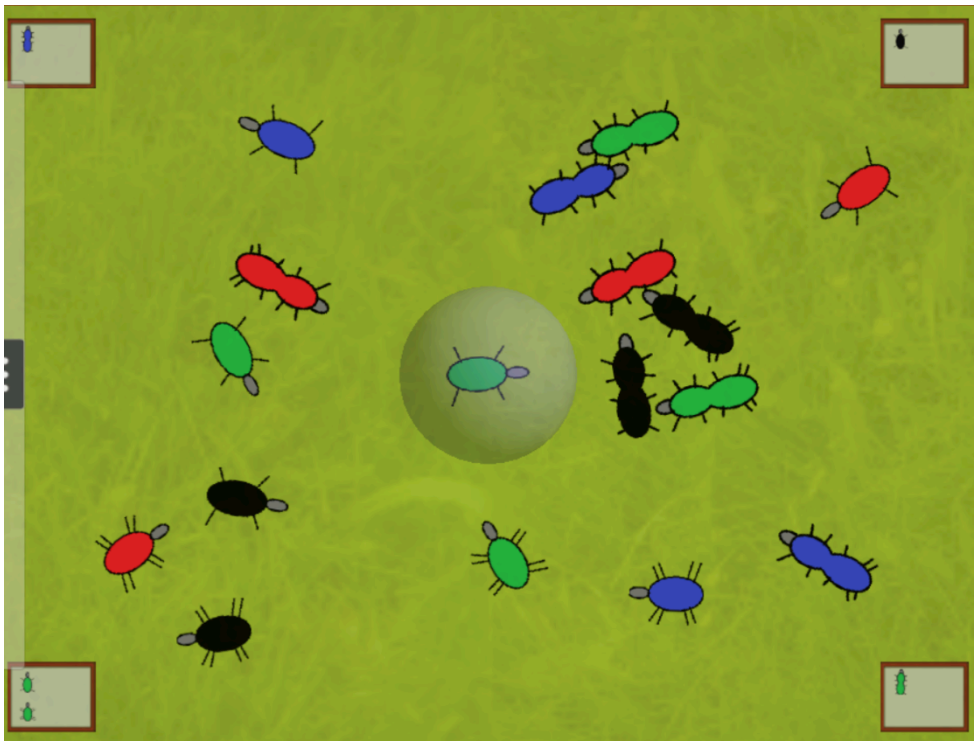

Figure 28. The first look for the bug game.

When a player drags the right bug to the player table it is added to the player table and a new bug is spawned randomly from the four spawn points around the game screen outside the view of the game. That bug then gets the target

position from the screen and eventually walks back to the game area for the players to see. At the same time the dome in the center containing the example bug is darkened so that players can't see the bug anymore and new bug is randomized as an example bug. After this the dome is lightened up again for players to see the example bug. The darkening and the lightening were done with Unity animator component. At this point the pause menu was also added to the game. The same pause menu was used for the bug game as was used for the card game but this time it was animated to slide in to view instead of just popping up on screen. This was also done using Unity's animator component.

5.3.4 Testing swipe to throw bugs

At this point a feature was tested that would allow players to swipe on top of the bugs and this would give the bugs force in the direction swiped and make the bugs fly across the screen. This feature turned out to be very hard to implement in a way that I would not break the bug pick up functionality. It was also hard to add just the right amount of force to the bugs thrown based on the swipe velocity. Many failed iterations were created of this feature until it work somewhat in the level desired but eventually it was decided to leave this feature out of the game and focus on the core mechanics.

5.3.5 Refining the touch script

As mentioned earlier in the card game section of the thesis the touch script was refined multiple times during the project and this was the first major point in which the refinements were made. The touch script was completely re-written from the ground up and the logic to call functions that touch events should trigger was thought again. After the refinements there was only one script that had touch events and every function that needed to be called with

touch events was called from that script. This way it was easy to maintain and add functions later. This also made the game a little lighter for the devices to run. The next major refinement to the touch script took place during the making of the 3D version of the prototypes that will be covered a little later in the thesis.

While the touch script was updated the bug images also received an update. Zaibatsu Interactive Graphic Designer Maria Pakkanen created a quick sprite sheet of new bug images to use in the project that were added to the game right away. This helped out in testing of the idea in practice because the bugs had smaller differences then the original bugs and the number of bugs was increased from 16 to 72. The new bug sprite sheet can be seen in Figure 29.

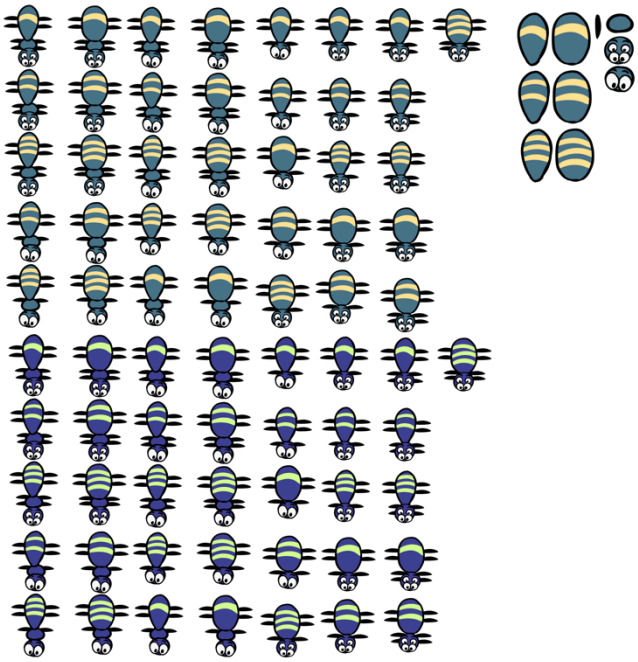

Figure 29. The new bug sprites.

5.3.6 Adding events

Now that the basic bug game was finished it was again time to add some flavor to the basic game. This time it was done in the form of events that could

happen randomly during the game. Each of the events could happen if no other events were currently on screen. The events were triggered using a timer, if enough time had passed an event was chosen at random if no other events were currently on screen.

The first of these events was the Leaf event where big leaves would sometimes be blown in front of the bugs by the wind. These leaves would then have to be moved by players in order to find the right bug from under the leaves. This event was done using Unity's animator component. The leaves are placed randomly to the right side of the screen starting from the edge of the screen and ending at the two times the screen width. When the Leaf event is called the animator is used to move all of the created leaves from the right to the middle of the screen and in front of the bugs and the game area. After players have moved the leaves around enough to find the right bug the leaves would then blow away again by the wind. Picture of the Leaf event is shown in Figure 30 below.

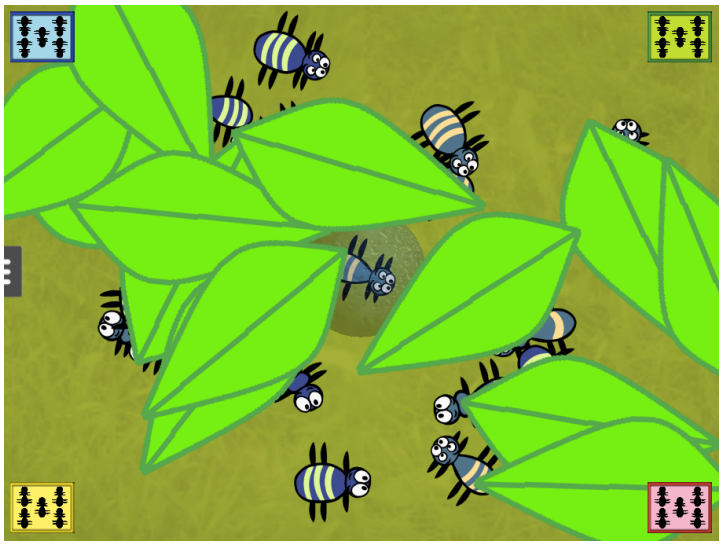

Figure 30. The Leaf event in action.

The second event was the Bubble event where bubbles start to float from the bottom and the top of the screen and players had to break the bubbles by tapping to drag the bugs underneath. The bubbles would not stop floating on

screen even if the players would manage to break every single one of the bubbles on screen. The new bubbles were created back as the old ones were broken. This was done using the update function call and the Vector2.MoveTowards method. The odd numbered bubbles would start floating from the bottom of the screen and the even numbered bubbles would start floating from the top of the screen at the random positions of the x-axes between the zero point and the screen width. If the bubble was broken or it reached the bottom or the top of the screen (depending on the starting position) then it would return to its starting position and start floating back to the screen. When the players find the right bug the bubbles would break. Figure 31 presents the Bubble event.

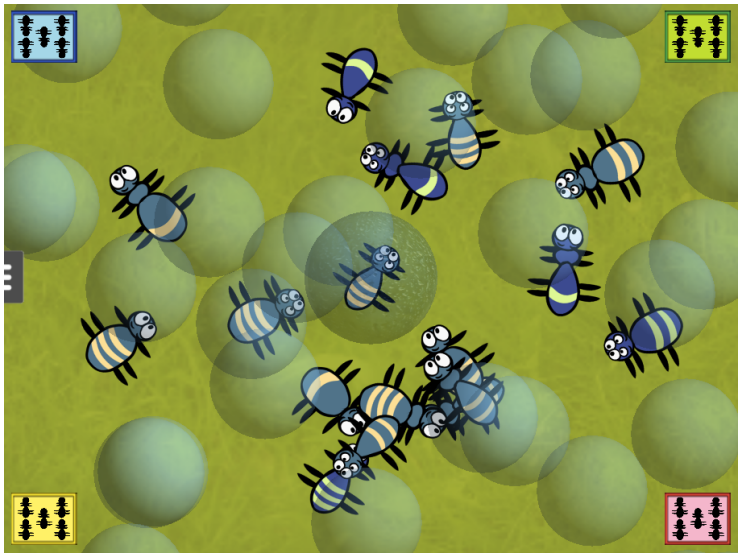

Figure 31. The Bubble event in action.

The last event created was the UFO event. In this event flying saucers would fly to the game screen randomly from every direction outside the view of the players trying to abduct the bugs players are attempting to find. If enough bugs were abducted the players lose. The players could break the UFO's by tapping on each one three times. If the UFOs manage to abduct a bug the bug would then be returned to the game area using one of the spawn points outside the devices screen. The UFOs have three states that indicate how

damaged they are, normal, alert and damaged. At first the UFOs are at the normal state and when they are tapped once they turn to the alert state and the UFO gets a slightly damaged look. With the next tap the UFO gets a damaged state and the look turns to red and badly damaged look. The UFOs work in a way that when the UFO event is called the UFOs start spawning from four different spawn points around the outside of the screen at random time intervals to each other. Each of the UFOs gets a bug as a target that is chosen randomly and the UFO then follows that bug until it is on top of it. After this the bug stops its movement and follows the UFO under it until the UFO is destroyed or it manages to abduct the bug. If the UFO is destroyed the bug takes a new random target and starts moving in that direction as it normally would and if UFO manages to abduct the bug it then spawns to a random spawn point and continues moving normally. Figure 32, the look of the UFOs is illustrated. In the picture there are three UFOs each showing the different damaged state, normal, alert and damaged.

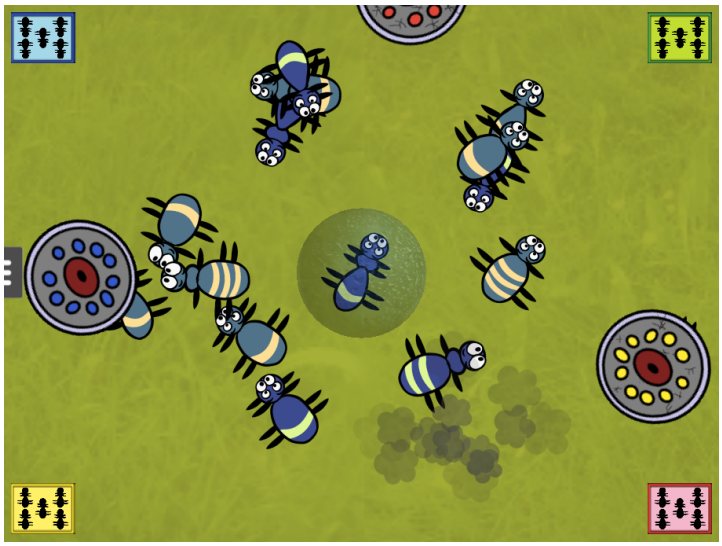

Figure 32. The UFO event in action.

When the event system was working the way it was intended the work on the next and the last prototype began. This was the 3D FPS style game that is described in detail below.

5.4 Finders Keepers

5.4.1 Starting the last prototype

The last prototype to create was intended to be the most different of the lot. This is why the 3D FPS idea for the final prototype was chosen. The basic idea was kept the same but everything else was changed. The last prototype was a 3D FPS game where players hunt for chess pieces around the level created by Ville Hyytiäinen and try to return them to the starting area before the other players can steal the piece. The game ends if either all of the chess pieces are returned in which case the player that had collected the most pieces wins or if one of the players collected more than half of the pieces in the level. The game had a starting point were players could look for the example that the players are supposed to find and the chess pieces were randomly placed through out the level. The players could pick up and carry objects such as the chess pieces or certain crates and also shoot other players with cannonballs.

The touch control script got its final change here. Now inside the touch script only calls to other scripts for look, shooting and pick up of the objects were used and the touch script only checks what kind of touch was happening on screen and at what position, nothing else. For example the player look around scripts look function is called from the touch script when the first touch on screen triggers the moved method of the Unity's TouchPhase class or the shooting scripts shoot function is called when a double tap is made.

5.4.2 The controls

The controls wanted for the last prototype had to be as simple as possible so that the game could still support 4 players in a single device. The basic

controls used for the game were that the first finger on screen would control the camera turning it in the direction dragged. For the movement second touch on screen was used so that if two fingers were touching the screen at the same time the player character would start moving forward. The object pick up was done using double taps. Double tapping in front of the object that could be carried would pick up the object and double tapping again would drop the carried object. The double tap was also used to shoot cannonballs if no objects were carried and no objects could be picked up.

The creation of the last prototype started with the writer of this thesis building a play scene were the controls, movement, picking up of objects, collisions and four-player split screen implementation could be tested while Ville Hyytiäinen began working on the creation of the level. The first thing to do was to create a player object that would turn around using the first touch and walk forward with the second. Unity's own mouse look script was used as a reference on creating the look around script. It was heavily modified and stripped to its bare bones to fit the need of the project. It could not be used as is because the four-player split screen support was added and the mouse look only worked for one player but mouse look script was used for testing purposes through out the development of the last prototype. The primitive capsule GameObject was used as a basis of the player character. The cube GameObject was also tested but the capsule GameObject proved to be a better choice for collisions and esthetics. The camera object was fitted as a child to the capsule object to represent the head of the player character and the look around script was added to both the capsule object and the camera object. The look around script used for the camera object controlled the y-axes movement of the look and the look around script used for the capsule object controlled the movement in x-axes. In Figure 33 a play scene can be seen where the first character controls and testing was done. It also shows the

difference between one-player game and the four-player game that is shown later in the thesis in Figure 35.

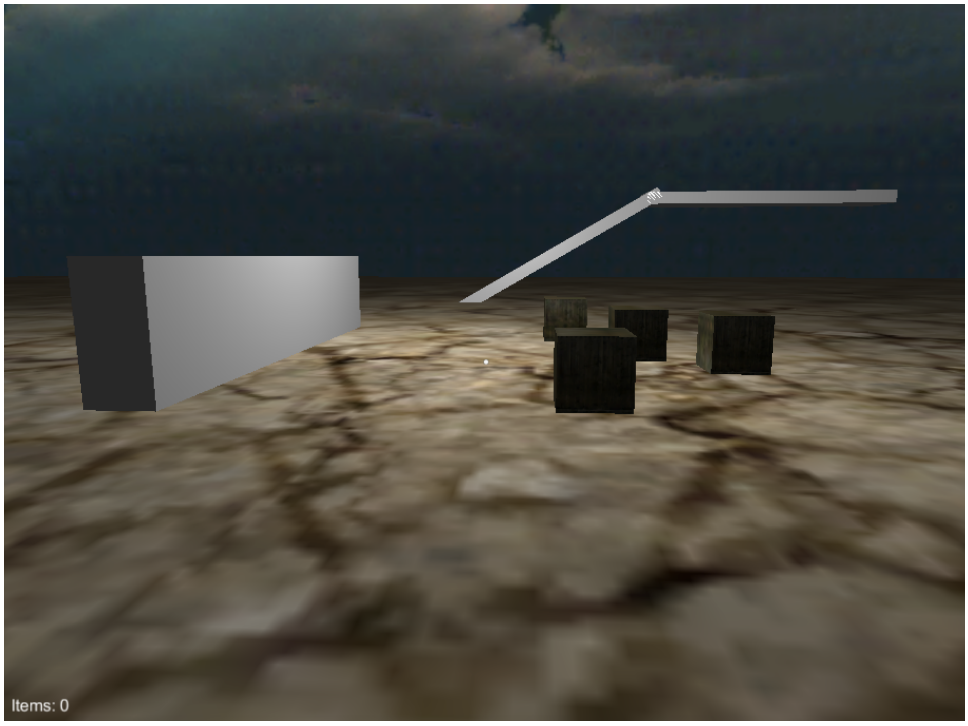

Figure 33. The test scene created to test the controls and collision.

After the character look had been finalized the movement and shooting was added. Ball kicking was also added at this phase but was later removed as a useless feature in this prototype. The movement of the character had to be a simple control method so it was decided that the players could only move forward and this should be done so that by placing the second finger on the screen would move the character forward. By testing it was also noted that the look around feature should not work with the second touch but only with the first so the only thing that the second touch was allowed to do was to move the character forward and nothing else. The shooting was added next so that when the player double tapped the screen a bullet would shoot out of the center of the screen with a certain amount of force. This was done by instantiating the bullet prefab in the center of the camera view and force was added to the Rigidbody of the bullet to shoot forward. The bullet prefab was

fitted with a collider and a Rigidbody components and a short script for destroying of the bullet.

5.4.3 The difficulty of the pick up

The simple hardest part of the character controls to implement was the object pick up and carry function. The way it was intended to work was that when player was close enough to the object that can be carried, double tapping would pick up the object and start floating it in front of the player while double tapping again would drop the object. As a matter of fact the first iteration of the pick up and carry feature was made so that tap and hold would pick up the object and then letting go of the touch would drop the picked object. This was later on changed so that simple double tap was used for both the pick up and drop of the object.

The object pick up was fairly simple to create by shooting a raycast from the center of the screen to the game world with a certain distance. If the raycast hit any of the objects that could be carried the object to be carried was set as a child of the player characters camera object and the Rigidbody of that object was set as kinematic. This made the object float in front of the players view and double tapping again removed the player's camera object as the parent of the pick up object and set the kinematic off from the Rigidbody so that the pick up object dropped back down to the floor. The problems arouse when the carried object would collide with other objects in the game world such as walls and floors while it was being carried. While the picked up object was set as kinematic it collided with other movable objects fine but colliding with other kinematic object such as walls and floor made the carried object go through them. This in turn was a problem because the player could drop the object inside walls and such and the carried object might be lost for the end of the game.

Different kinds of versions of the pick up were tested that included a version were a spring was added between the player camera object and the carried object so that the carried object could have bounced in front of the players view when it collided but in turn this made it so that the collision bounced the player character away from walls and such so it was removed. Another version was were an invisible point was added in front of the player that the carried object was set to follow but this again did not work as it was planned and was scrapped soon after testing.

Finally it was decided that the pick up and carry function should work the way it was first created and extra colliders were added to the game world in the case of the players trying to drop the carried object behind the walls of the play area. This worked well enough for the prototype to continue creating other aspects of the game. The player carrying an object is shown in Figure 34. Also in this figure the final level can be seen and the return area where the example piece is located and the found pieces should be returned.

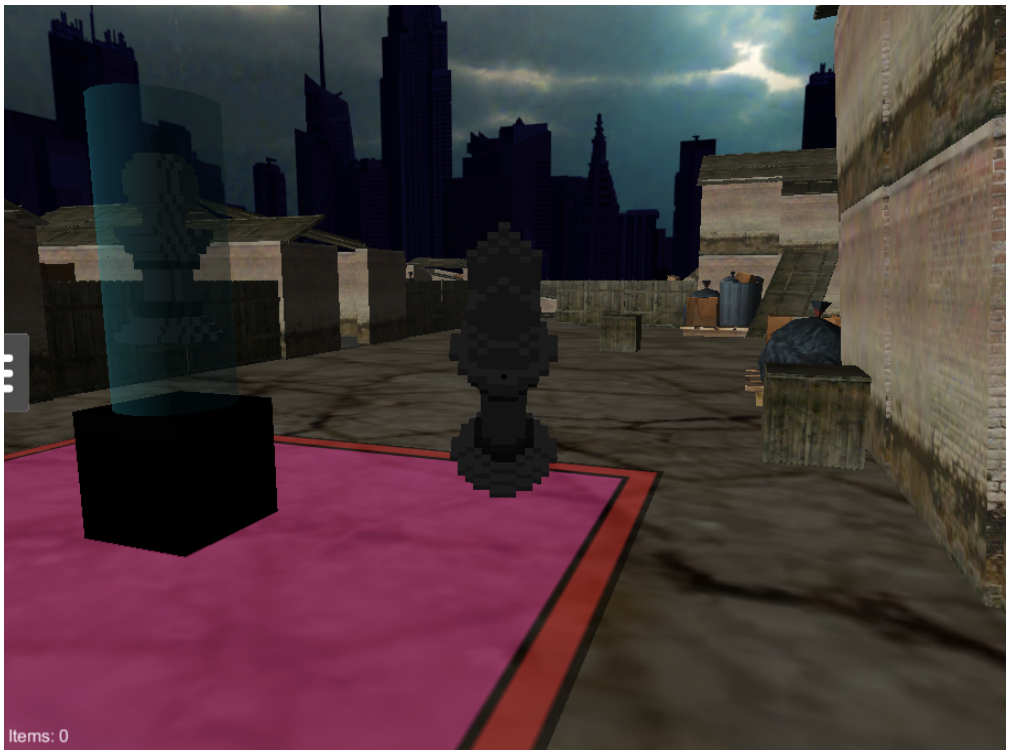

Figure 34. The player carrying a chess piece.

5.4.4 The four-player split screen

When the controls for the single player game worked the way they were designed it was time to create the up to four-player splits screen feature. The screen was divided into four views, one for each player in the game. Four of the player characters where added to the scene and tagged accordingly by their color, player 1 was red, player 2 was green, player 3 was blue and player 4 was yellow. Then each of the character camera views were resized to accommodate a small portion of the screen. At this point it was clear that the views for two of the players had to be flipped so that the half of the players could play from the other side of the device and still have their image the right way. For this a small script was used to flip the cameras upside down for the second and fourth players.

After the screen was set up for a four-player game it was noticed that the controls would have to be changed a little because at this stage a single player still controlled all the player characters in the screen. This was so because there was no definition in the touch script of on which players screen the touch was happening. A check was added to the touch script to check if touch was happening in a certain characters camera view and allow the movement, shooting and all the other touch calls to happen only when they happen inside that characters view. Figure 35 illustrates the picture of the four-player screen.

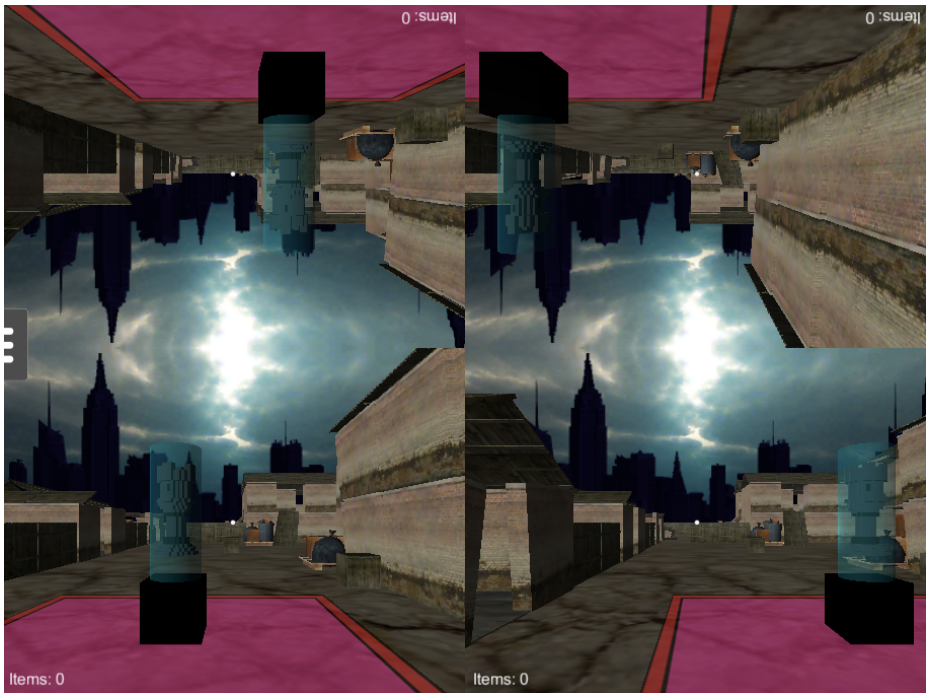

Figure 35. The four-player splitscreen.

5.4.5 Putting the pieces together

When the four-player view was finished it was time to put the pieces together and test the character controls in the level created by Ville Hyytiäinen. The level contained spawn points for the chess pieces where they were spawned randomly at the beginning of the game. The level also contained the return area to return the right chess piece and where the players could also look at the example chess piece that they were currently trying to find. Finally in the level were some houses and house ruins and a city landscape to add to the immersion. Adding the player characters to the level happened smoothly and the game could be play tested soon after. At this point the pick up and carry objects was finalized to the form it was in the final prototype. The way to pick up the objects was changed to double tap and some info text were added to inform the player when they can pick up the objects. This was done by constantly raycasting from the center of the screen to see if the player was close enough to an object that could be picked up and if so a text message

informed the player that double tapping would pick up this object. The shooting was also done using the double tap but it could only happen if the player was not carrying an object or if there was no object to pick up.

The info texts were also added to inform the players when one of the players had picked up the right chess piece or brought it to the return area. In addition to that there was also a crosshair created for the players to better see where they were aiming and a score text to see how many chess pieces the player has returned. The final prototype was finalized by adding the same kind of menus and player select screens as were used in the bug and the card game prototypes. Figure 36 illustrates the info texts when the player is picking up the right chess piece. The two-player layout is also shown in this figure.

Figure 36. The info texts and the two-player mode.

5.5 Summary

From all the versions created the last prototype was definitely the most refined of the bunch. The touch script was optimized in the best way and the idea behind scripting in Unity had become clearer to the writer of this thesis. The scripts in the last prototype were much smaller and much more focused on the specific tasks then in the previous two prototypes.

All of the prototypes created had their purpose and each had their own catch. The card game was the straight up port of the original board game that started the idea for this project. It shows how the basic idea could work if it was converted to the mobile platforms without changing much of anything. The next was the bug game, which had the same feel and idea of the original board game; however it added features like the UFOs that would not be possible in the board game form. It also added the movement for the “cards” in the form of the bugs that also added an extra spice to the game, which could not be done (not so simply at least) in the original game. The final version of the prototypes created was also the most “out there” version of the games. The idea for the game was to make this as different as possible but still keeping in mind the original idea. The game is still about finding the right object from all the other objects while comparing it to the example object but done in a way that does not look at all like the original game. It was also the only 3D game of the three that also brought a new aspect to the game.

The game was play tested by the students of the Game Development course at JAMK and based on the feedback the version of the game liked most was the bug game. It was said to be the easiest to pickup and play and the idea of what to do in the game was clearer than in the card game. It is to be noted that the last prototype was not tested at that time because it was not yet complete.

6 CONCLUSIONS

My final thoughts on the project are extremely positive. I enjoyed working with Unity and with games in general and I am very glad that Zaibatsu Interactive offered this opportunity to me. The work we did with Ville Hyytiäinen was fun and rewarding; however at times also extremely challenging. It was also great to have so much creative freedom when we were building the prototypes. Jussi Perttola at Zaibatsu Interactive trusted us a great deal and gave us the freedom to choose how we approached the task at hand but the entire team at Zaibatsu Interactive also provided us plenty of feedback and help if and when it was needed.

All three of the games provided their own unique challenges and very little of the same code was used between those games although especially the first two were very similar in nature. This is because I feel like I learned a great deal in each and every one of those prototypes and managed to find better, more efficient ways to produce the same things. I sometimes wanted to challenge myself by making things differently on purpose to see how things worked in Unity. The animator component for example was not necessary for the project but I wanted to use it a lot because I wanted to learn how it worked and I think I managed to do just that.

I wanted to use the prototyping of video games as one of the theory sections for this thesis but when searching for materials concerning prototyping there was very little to find and I was slightly surprised by that. I think that this could be an extremely good place to make a thesis of how different gaming companies do prototyping on their ideas. Of course this could also contain software companies in general if the writer of such a thesis preferred that.

As written in the theory section of the prototyping the prototyping should be a task where small mechanics and aspects of the game are prototyped; however this was done slightly differently with our prototypes because we were prototyping an idea of a game and not one specific mechanic. What we created were games that could be played from start to finish and had all the mechanics and aspects that are normally seen in the game like all the menus and a game that had a start and a finish and the object of the game was clear.

When I was making the touch script for the games and it changed and evolved a great deal during the project I started wondering that why there is not a touch script component build straight into Unity. Is this not a fundamental component when developing games to mobile platforms and is Unity not trying to be “the do all be all” of creating games? I think this is something that Unity should definitely add to the Unity engine. With this touch script I mean that the component could detect different types of touches such as taps, double taps, tap and holds or different kind of swipes, long and short, fast or slows. This would be a very useful addition and I think it would save plenty of time for many of the developers when making their games. Fortunately the Unity Asset store is filled with touch scripts that do just these tap and swipe gestures I was personally missing from Unity but I still think this should be a build in component.

Otherwise the Unity game engine was a joy to use and when you got the hang of the logic behind creating games with Unity it was and immensely simple, easy and efficient to use. You could do a lot of neat things with it with just a little effort. I would definitely like to work in a commercial game development project that uses Unity as the main tool.

In my own opinion the prototypes that we created with Ville Hyytiäinen were quite successful. All of the three prototypes were different and had different approaches to the same idea. I was surprised at how quickly we managed to

build the first prototype considering that I was quite new to using Unity and had not used it much at all before starting this thesis project. My knowledge involved one game development course where I was tasked to building a 3D menu for the game and of a LevelUp day where we build a simple space shooter in one day with the help of a Unity expert. The first prototype we build was finished in around a month and plenty of the time within that month was used fixing bugs and fine-tuning the game. I'd say the core gameplay was ready in little over two weeks.

The prototype that I enjoyed building the most was the final prototype that was the only 3D game of the prototypes created. One of the biggest reasons of why I liked building that particular game was probably that I had gotten the hang of the Unity engine the best at that time. The biggest thing that I was not pleased with in the project from the start was the touch script I created for the first prototype and finally when starting the last prototype I managed to create the touch script that I could be proud of. Also, it was interesting to do a 3D prototype after the first two prototypes were 2D games because it was so different. It was also fun and challenging to see how an FPS style game could work in a mobile platform.

Of the three prototypes created the one that I think should and could be developed further is the bug game. It was the most liked prototype when the students of the game development course tested it and it is in my opinion the most accessible of the prototypes. The idea of the game is simple enough to start playing right away and the events, if developed further, are a fun addition and a good way to change the flow of the game. Especially the UFO event proved to be a lot of fun and this event could be used so that it was in a more dominant role in the game. For example there could be a mothership for the UFOs that float in front of the playing area and it would spawn new UFOs to

the game and players would have to work together to destroy the mothership and the spawning UFOs to continue finding bugs.

The least favorite game for me was the card game because it did not add anything new to the idea and the basic idea did not work as well in the mobile platform as it did as a board game. Then again the 3D game was fun to play and develop but it was somewhat too massive of a game to put in a mobile screen especially if the game had to contain a four-player split screen mode.

One of the problems with searching to the theoretical side of the thesis was the aforementioned lag of information on prototyping video games but also the fact that writing about Unity the best source of information was the Unity documentation itself. That of course is the most reliable place to look for answers concerning Unity but when the thesis should have a lot of different sources that the information is gathered from that caused a dilemma for me. Although I think the most accurate info on Unity can be found from Unity page itself I used a few other sources to get the information to add some variety to the writing.

Working together with Ville Hyytiäinen went really well in my opinion and I felt like the workload was divided equally among us. Ville worked with design aspects of the prototypes and thought of new ideas that could be tested and I did my best to make those ideas work. Ville also made the level for the last game, which in my opinion shows that he has some great talent as a level designer. It was a good-looking level with multiple floors, rooms and buildings. Ville also made a lot of graphics for the first two prototypes what included bug, leaf and UFO sprites. In turn I had my hands full with scripting and making the ideas work as intended.

As is mentioned earlier in this thesis the writing process was done individually. In this aspect I think it could have gone better. With this I mean that we should

probably have written at least the first chapter completely together. But I think we did this the way we did because we felt like we could get more done quicker if we both concentrated on our own writing. We did however have a one afternoon cooperation for the writing process at the end of March where we discussed how we would write the first chapters to our thesis's. But overall I enjoyed my time working with Ville and think he did a tremendous job with his thesis work.

What I learned from this project was that Unity is an excellent tool for creating games especially for the beginners. It is flexible so that it can easily be used to create 2D or 3D games. I also learned to use Unity much better than I previously could and some aspects, e.g. the animator component are now quite clear to me. Since I wrote the scripts using C# I also got to hone my skills in that language as well. I have already used C# a lot before this project but using it in this kind of a project got me better in new areas of C# language. I also learned to work in a game development project because we created the games together with Ville Hyytiäinen who was responsible for the design aspects while I was responsible of the programming aspects. And while we worked on the projects I also learned to use the communication and project management tools such as Trello and Flowdock. Although Ville and I did not use Flowdock much during work because we worked side-by-side and verbal communication was the best way to communicate, I did use it while at home and while contacting the Zaibatsu Interactive team. I learned that Trello was a great project management tool to keep track of what task should be done next and who was doing what at any specific time.

As a final note I have to say I am really clad to have gotten this opportunity to make this thesis for the gaming industry.

References

2D (Concept). N.d. Giant Bomb article defining 2D. Accessed on 3 April 2015. http://www.giantbomb.com/2d/3015-1427/

About Flowdock. N.d. Flowdock homepage. About Flowdock. Accessed on 2 April 2015. Retrieved from https://www.flowdock.com/about

Android, the world's most popular mobile platform. N.d. Android homepage on what is Android. Accessed on 3 April 2015. Retrieved from http://developer.android.com/about/index.html

Apple – iOS 8 – What is iOS? N.d. Apple homepage on what is iOS. Accessed on 3 April 2015. Retrieved from http://www.apple.com/ios/what-is/

Apple – iPad. N.d. Apple product page on iPad. Accessed on 9 April 2015. Retrieved from https://www.apple.com/ipad/

Apple – iPhone. N.d. Apple product page on iPhone. Accessed on 9 April 2015. Retrieved from https://www.apple.com/iphone/

Apple – iPod. N.d. Apple product page on iPod. Accessed on 9 April 2015. Retrieved from https://www.apple.com/ipod/

Baldur's Gate –Wikipedia. 2015. Wikipedia page on Baldur's Gate. Accessed on 9 April 2015. Retrieved from http://en.wikipedia.org/wiki/Baldur%27s_Gate

Blackman, S. 2013. Beginning 3D Game Development with Unity 4. Second Edition. Apress.

Deus Ex – Wikipedia. 2015. Wikipedia page on Baldur's Gate. Accessed on 9 April 2015. Retrived from http://en.wikipedia.org/wiki/Deus_Ex

Diablo III – Wikipedia. 2015. Wikipedia page on Diablo III. Accessed on 9 April 2015. Retrived from http://en.wikipedia.org/wiki/Diablo_III

Dropbox - About Dropbox. N.d. Dropbox homepage. About Dropbox. Accessed on 2 April 2015. Retrieved from https://www.dropbox.com/about

Dungeons and Dragons. 2015. Wikipedia page on Dungeons and Dragons. Accessed on 9 April 2015. Retrived from http://en.wikipedia.org/wiki/Dungeons_%26_Dragons

Epic Mickey – Wikipedia. 2015. Wikipedia page on Epic Mickey. Accessed on 9 April 2015. Retrived from http://en.wikipedia.org/wiki/Epic_Mickey

Fullerton, T. 2014. Game Design Workshop. 3rd Edition. CRC Press.

Git. N.d. Git homepage. Accessed on 3 April 2015. Retrieved from http://git-scm.com/

Hyytiäinen, V. 2015. Digitalizing a board game concept. Mobile game design and development. Bachelor's thesis. JAMK University of Applied Sciences, The School of Technology, Degree Programme in Software Engineering. To be submitted to https://www.theseus.fi/ in May 2015.

Janssen, C. N.d. What is First Person Shooter (FPS)? – Definition from Techopedia. Techopedia webpage. Accessed on 3 April 2015. Retrieved from http://www.techopedia.com/definition/241/first-person-shooter-fps

Janssen, C. N.d. What is Integrated Development Environment? Techopedia definition for IDE. Accessed on 2 April 2015. Retrieved from http://www.techopedia.com/definition/26860/integrated-development-environment-ide

Janssen, C. N.d. What is script? Definition from Techopedia. Accessed on 3 April 2015. Retrieved from http://www.techopedia.com/definition/10324/scripts

Janssen, C. N.d. What is Visual Studio Express? Techopedia definition for Visual Studio Express. Accessed on 2 April 2015. Retrieved from https://www.visualstudio.com/

Katamari Damacy – Wikipedia. 2014. Wikipedia page on Katamari Damacy. Accessed on 9 April 2015. Retrived from http://en.wikipedia.org/wiki/Katamari_Damacy

Nordgren, K. 2014. Peliteollisuus – kehityspolku. Study made by Tekes about the rise of Finnish game industry. Accessed on 3 April 2015. Retrieved from http://www.tekes.fi/en/whats-going-on/news/the-rise-of-the-finnish-game-industry/

Menard, M. 2012. Game Development with Unity. Course Technology.

Pikachu – Wikipedia. 2015. Wikipedia page on Pikachu character. Accessed on 3 April 2015. Retrieved from http://en.wikipedia.org/wiki/Pikachu

PS4 – Features, Games and Videos. N.d. Playstation 4 us homepage, overview. Accessed on 3 April 2015. Retrieved from http://www.playstation.com/en-us/explore/ps4/

Rouse, M. 2007. What is C#?. TechTarget Definition of C#. Accessed on 2 April 2015. Retrieved from http://searchwindevelopment.techtarget.com/definition/C

Slick, J. N.d. 3D Defined – What is 3D? About Tech. Accessed on 3 April 2015. Retrieved from http://3d.about.com/od/3d-101-The-Basics/a/3d-Defined-What-Is-3d.htm

Spore – Wikipedia. 2015. Wikipedia page on Spore. Accessed on 9 April 2015. Retrieved from http://en.wikipedia.org/wiki/Spore_%282008_video_game%29

Texture definition, N.d. Definition of the word Texture from the Dictionary.com. Accessed on 6 April 2015. Retrieved from http://dictionary.reference.com/browse/texture

Unity - Fast Facts. N.d. Unity homepage company facts. Accessed on 1 April 2015. Retrieved from https://unity3d.com/public-relations

Unity – Game engine, tools and multiplatform. N.d. Unity homepage. Accessed on 1 April 2015. Retrieved from http://unity3d.com/unity

Unity – Manual: Colliders. N.d. Unity documentation on Colliders. Accessed on 9 April 2015. Retrieved from http://docs.unity3d.com/Manual/CollidersOverview.html

Unity – Manual: GameObjects N.d. Unity documentation on GameObjects. Accessed on 6 April 2015. Retrieved from http://docs.unity3d.com/Manual/GameObjects.html

Unity – Manual: Learning the Interface, N.d. Unity documentation on learning the interface. Accessed on 3 April 2015. Retrieved from http://docs.unity3d.com/Manual/LearningtheInterface.html

Unity – Manual: MonoDevelop, N.d. Unity documentation on MonoDevelop. Accessed on 3 April 2015. Retrieved from http://docs.unity3d.com/Manual/MonoDevelop.html

Unity – Manual: Prefabs N.d. Unity documentation on Prefabs. Accessed on 6 April 2015. Retrieved from http://docs.unity3d.com/Manual/Prefabs.html

Unity – Manual: Project Browser, N.d. Unity documentation on Project Browser. Accessed on 6 April 2015. Retrieved from http://docs.unity3d.com/Manual/ProjectView.html

Unity – Manual: Rigidbodies. N.d. Unity documentation on Rigidbodies. Accessed on 9 April 2015. Retrieved from http://docs.unity3d.com/Manual/RigidbodiesOverview.html

Unity – Manual: Tags. N.d. Unity documentation on Tags. Accessed on 9 April 2015. Retrieved from http://docs.unity3d.com/Manual/Tags.html

Unity – Manual: Toolbar N.d. Unity documentation on GameObjects. Accessed on 6 April 2015. Retrieved from http://docs.unity3d.com/Manual/Toolbar.html

Unity – Manual: Visual Studio C# Integration, N.d. Unity documentation on Visual Studio C# Integration. Accessed on 3 April 2015. Retrieved from http://docs.unity3d.com/Manual/VisualStudioIntegration.html

Unity – Scripting API: Physics2D.Raycast. N.d. Unity documentation on Raycasting. Accessed on 9 April 2015. Retrieved from http://docs.unity3d.com/ScriptReference/Physics2D.Raycast.html

Warren Spector – Wikipedia. 2015. Wikipedia page on Warren Spector. Accessed on 8 April 2015. Retrived from http://en.wikipedia.org/wiki/Warren_Spector

What is Trello? – Trello Help. 2015. Trello help page. What is Trello? Accessed on 2 April 2015. Retrieved from http://help.trello.com/article/708-what-is-trello

Xbox One – Official Site. N.d. Xbox One us homepage. Accessed on 3 April 2015. Retrieved from http://www.xbox.com/en-US/xbox-one

Zaibatsu Interactive - About Us. N.d. Zaibatsu Interactive homepage about us. Accessed on 1 April 2015. Retrieved from http://zaibatsu.fi/about/

Zaibatsu Interactive – Elder Goo. N.d. Zaibatsu Interactive homepage Elder Goo. Accessed on 3 April 2015. Retrieved from http://zaibatsu.fi/eldergoo/